Learning Submodular Mixtures; and Active/Semi-Supervised Learning

Jeff Bilmes

Department of Electrical Engineering
University of Washington, Seattle
http://ssli.ee.washington.edu/~bilmes

Wednesday, March 21st, 2012

Summary

- Submodular functions are finding ever more application in machine learning.
- They naturally represent and successfully solve the problem of document summarization.
- They can be used to parameterize a class of joint active/semi-supervised learning algorithms.
- A need for both fast submodular function maximization and minimization on large ground set sizes.

Acknowledgments

Joint work with my students:

Hui Lin

Andrew Guillory

Outline

Outline

Outline

Document Summarization

- Proliferation of data and documents.

Document Summarization

- Proliferation of data and documents.
- Document Summarization: Given a large collection of text documents, produce a short human-readable summary that accurately represents the documents.

Document Summarization

- Proliferation of data and documents.
- Document Summarization: Given a large collection of text documents, produce a short human-readable summary that accurately represents the documents.
- Extractive Document Summarization: The summary is comprised of parts of the original.

Document Summarization

- Proliferation of data and documents.
- Document Summarization: Given a large collection of text documents, produce a short human-readable summary that accurately represents the documents.
- Extractive Document Summarization: The summary is comprised of parts of the original.
- E.g., if V is the set of all sentences in the documents, then $S \subset V$ is a candidate summary.

Document Summarization

- Proliferation of data and documents.
- Document Summarization: Given a large collection of text documents, produce a short human-readable summary that accurately represents the documents.
- Extractive Document Summarization: The summary is comprised of parts of the original.
- E.g., if V is the set of all sentences in the documents, then $S \subset V$ is a candidate summary.
- If V is the set of all web pages, then $S \subset V$ is a candidate summary.

Document Summarization in Natural Language Processing

- $\exists$ many well-established methods for document summarization

Document Summarization in Natural Language Processing

- $\exists$ many well-established methods for document summarization
- Research in the natural language processing community have repeatedly but unknowingly used submodularity:

Document Summarization in Natural Language Processing

- $\exists$ many well-established methods for document summarization
- Research in the natural language processing community have repeatedly but unknowingly used submodularity:
  - Carbonell & Goldstein, 1998;
  - Filatova & Hatzivassiloglou, 2004;
  - McDonald, 2007;
  - Takamura & Okumura, 2009;
  - Riedhammer et al., 2010;
  - Shen & Li, 2010
  - Berg-Kirkpatrick et al., 2011

Document Summarization in Natural Language Processing

- $\exists$ many well-established methods for document summarization
- Research in the natural language processing community have repeatedly but unknowingly used submodularity:
  - Carbonell & Goldstein, 1998;
  - Filatova & Hatzivassiloglou, 2004;
  - McDonald, 2007;
  - Takamura & Okumura, 2009;
  - Riedhammer et al., 2010;
  - Shen & Li, 2010
  - Berg-Kirkpatrick et al., 2011
- These researchers did not intentionally use submodularity, and they did not directly use submodular optimization.

Document Summarization in Natural Language Processing

- $\exists$ many well-established methods for document summarization
- Research in the natural language processing community have repeatedly but unknowingly used submodularity:
  - Carbonell & Goldstein, 1998;
  - Filatova & Hatzivassiloglou, 2004;
  - McDonald, 2007;
  - Takamura & Okumura, 2009;
  - Riedhammer et al., 2010;
  - Shen & Li, 2010
  - Berg-Kirkpatrick et al., 2011
- These researchers did not intentionally use submodularity, and they did not directly use submodular optimization.
- Occasionally, the greedy algorithm was used for optimization (e.g., Carbonell & Goldstein, 1998).

Document Summarization in Natural Language Processing

- $\exists$ many well-established methods for document summarization
- Research in the natural language processing community have repeatedly but unknowingly used submodularity:
  - Carbonell & Goldstein, 1998;
  - Filatova & Hatzivassiloglou, 2004;
  - McDonald, 2007;
  - Takamura & Okumura, 2009;
  - Riedhammer et al., 2010;
  - Shen & Li, 2010
  - Berg-Kirkpatrick et al., 2011
- These researchers did not intentionally use submodularity, and they did not directly use submodular optimization.
- Occasionally, the greedy algorithm was used for optimization (e.g., Carbonell & Goldstein, 1998).
- The majority of researchers defined heuristics involving non-submodular objectives, and either greedy or ILP optimization strategies.

Document Summarization in Natural Language Processing: Evaluation

- Evaluating a summary requires human judgment.

Document Summarization in Natural Language Processing: Evaluation

- Evaluating a summary requires human judgment.
- $\exists$ standard methods to automatically evaluate a summary, to speed development time.

Document Summarization in Natural Language Processing: Evaluation

- Evaluating a summary requires human judgment.
- $\exists$ standard methods to automatically evaluate a summary, to speed development time.
- Standard methods to automatically evaluate the quality of a given summary are also (unwittingly) submodular:

Document Summarization in Natural Language Processing: Evaluation

- Evaluating a summary requires human judgment.
- $\exists$ standard methods to automatically evaluate a summary, to speed development time.
- Standard methods to automatically evaluate the quality of a given summary are also (unwittingly) submodular:
  - ROUGE-N (Lin, 2004);

Document Summarization in Natural Language Processing: Evaluation

- Evaluating a summary requires human judgment.
- $\exists$ standard methods to automatically evaluate a summary, to speed development time.
- Standard methods to automatically evaluate the quality of a given summary are also (unwittingly) submodular:
  - ROUGE-N (Lin, 2004);
  - Pyramid (Passonneau et al., 2005);

Document Summarization in Natural Language Processing: Evaluation

- Evaluating a summary requires human judgment.
- $\exists$ standard methods to automatically evaluate a summary, to speed development time.
- Standard methods to automatically evaluate the quality of a given summary are also (unwittingly) submodular:
  - ROUGE-N (Lin, 2004);
  - Pyramid (Passonneau et al., 2005);
- Again, these researchers did not intentionally use submodularity, and the objectives are often not monotone.

Document Summarization in Natural Language Processing: Evaluation

- Evaluating a summary requires human judgment.
- $\exists$ standard methods to automatically evaluate a summary, to speed development time.
- Standard methods to automatically evaluate the quality of a given summary are also (unwittingly) submodular:
  - ROUGE-N (Lin, 2004);
  - Pyramid (Passonneau et al., 2005);
- Again, these researchers did not intentionally use submodularity, and the objectives are often not monotone.
- The objectives, for a given set of documents, are parameterized by actual summaries created by humans so they are not available for optimization in practice.

Document Summarization in Natural Language Processing: Evaluation

- Evaluating a summary requires human judgment.
- $\exists$ standard methods to automatically evaluate a summary, to speed development time.
- Standard methods to automatically evaluate the quality of a given summary are also (unwittingly) submodular:
  - ROUGE-N (Lin, 2004);
  - Pyramid (Passonneau et al., 2005);
- Again, these researchers did not intentionally use submodularity, and the objectives are often not monotone.
- The objectives, for a given set of documents, are parameterized by actual summaries created by humans so they are not available for optimization in practice.
- They are only available for evaluation, and they correlate quite well with manual human-generated summaries

NIST's ROUGE-N evaluation function

NIST's ROUGE-N score is the standard evaluation measure, and it is polymatroidal:

$$f_{\text{ROUGE-N}}(S) \triangleq \frac{\sum_{i=1}^{K} \sum_{e \in R_i} \min(c_e(S), r_{e,i})}{\sum_{i=1}^{K} \sum_{e \in R_i} r_{e,i}},$$

where

- S is the candidate summary (a set of sentences extracted from the ground set V)

NIST's ROUGE-N evaluation function

NIST's ROUGE-N score is the standard evaluation measure, and it is polymatroidal:

$$f_{\text{ROUGE-N}}(S) \triangleq \frac{\sum_{i=1}^{K} \sum_{e \in R_i} \min(c_e(S), r_{e,i})}{\sum_{i=1}^{K} \sum_{e \in R_i} r_{e,i}},$$

where

- S is the candidate summary (a set of sentences extracted from the ground set V)
- $c_e : 2^V \rightarrow \mathbb{Z}_+$ is the number of times an n-gram e occurs in summary S, clearly a modular function for each e.

NIST's ROUGE-N evaluation function

NIST's ROUGE-N score is the standard evaluation measure, and it is polymatroidal:

$$f_{\text{ROUGE-N}}(S) \triangleq \frac{\sum_{i=1}^{K} \sum_{e \in R_i} \min(c_e(S), r_{e,i})}{\sum_{i=1}^{K} \sum_{e \in R_i} r_{e,i}},$$

where

- S is the candidate summary (a set of sentences extracted from the ground set V)
- $c_e : 2^V \rightarrow \mathbb{Z}_+$ is the number of times an n-gram e occurs in summary S, clearly a modular function for each e.
- R_i is the set of n-grams contained in the reference summary i (given K reference summaries).

NIST's ROUGE-N evaluation function

NIST's ROUGE-N score is the standard evaluation measure, and it is polymatroidal:

$$f_{\text{ROUGE-N}}(S) \triangleq \frac{\sum_{i=1}^{K} \sum_{e \in R_i} \min(c_e(S), r_{e,i})}{\sum_{i=1}^{K} \sum_{e \in R_i} r_{e,i}},$$

where

- S is the candidate summary (a set of sentences extracted from the ground set V)
- $c_e : 2^V \rightarrow \mathbb{Z}_+$ is the number of times an n-gram e occurs in summary S, clearly a modular function for each e.
- R_i is the set of n-grams contained in the reference summary i (given K reference summaries).
- and $r_{e,i}$ is the number of times n-gram e occurs in reference summary i.

NIST's ROUGE-N evaluation function

NIST's ROUGE-N score is the standard evaluation measure, and it is polymatroidal:

$$f_{\text{ROUGE-N}}(S) \triangleq \frac{\sum_{i=1}^{K} \sum_{e \in R_i} \min(c_e(S), r_{e,i})}{\sum_{i=1}^{K} \sum_{e \in R_i} r_{e,i}},$$

where

- S is the candidate summary (a set of sentences extracted from the ground set V)
- $c_e : 2^V \rightarrow \mathbb{Z}_+$ is the number of times an n-gram e occurs in summary S, clearly a modular function for each e.
- R_i is the set of n-grams contained in the reference summary i (given K reference summaries).
- and $r_{e,i}$ is the number of times n-gram e occurs in reference summary i.
- Note again, ROUGE-N is unavailable to optimize directly.

Extractive Document Summarization

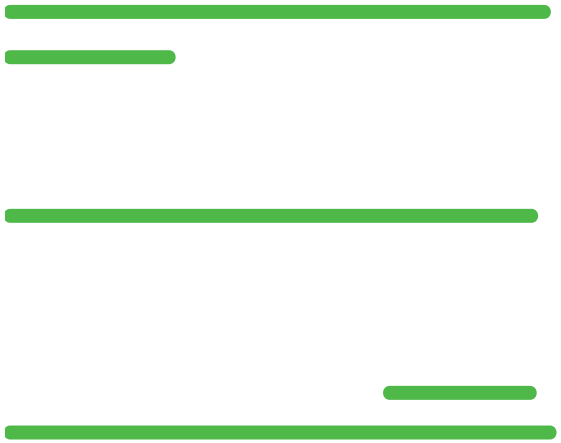
- The figure below represents the sentences of a document

Extractive Document Summarization

- We extract sentences (green) as a summary of the full document

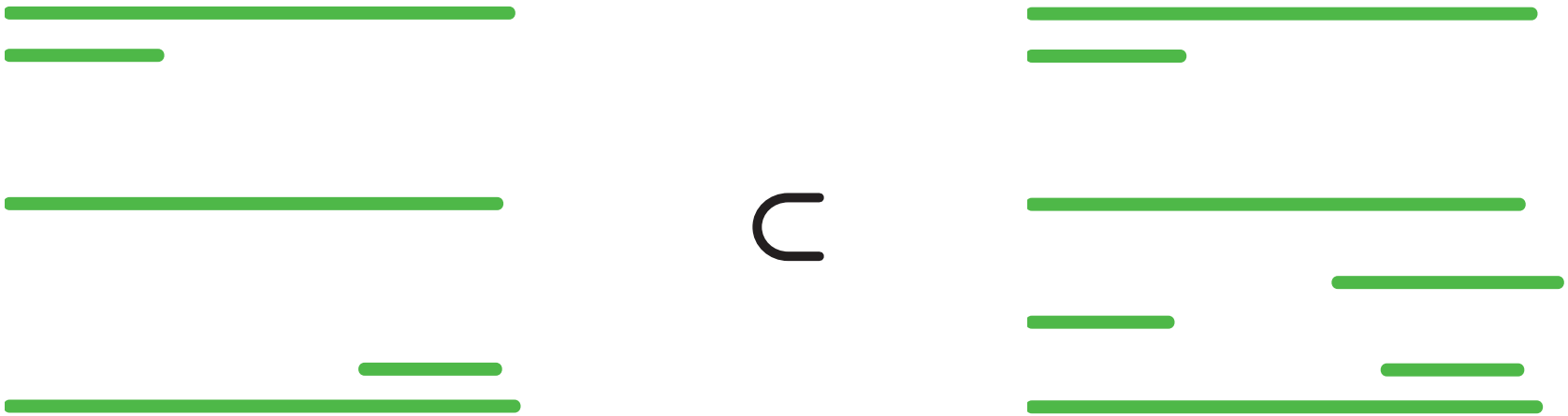
Extractive Document Summarization

- We extract sentences (green) as a summary of the full document



Extractive Document Summarization

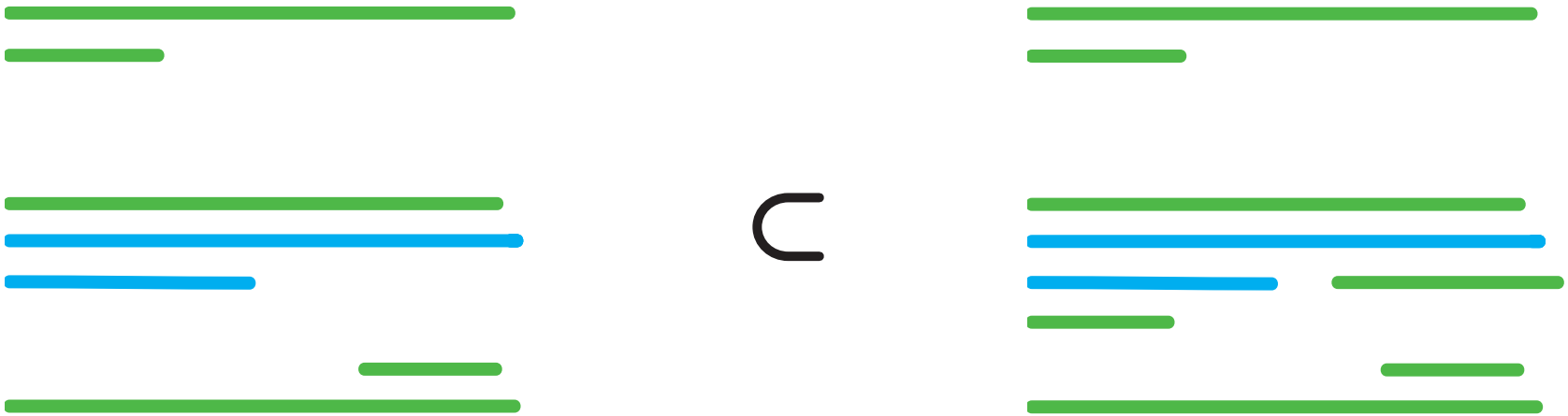
- We extract sentences (green) as a summary of the full document



- The summary on the left is a subset of the summary on the right.

Extractive Document Summarization

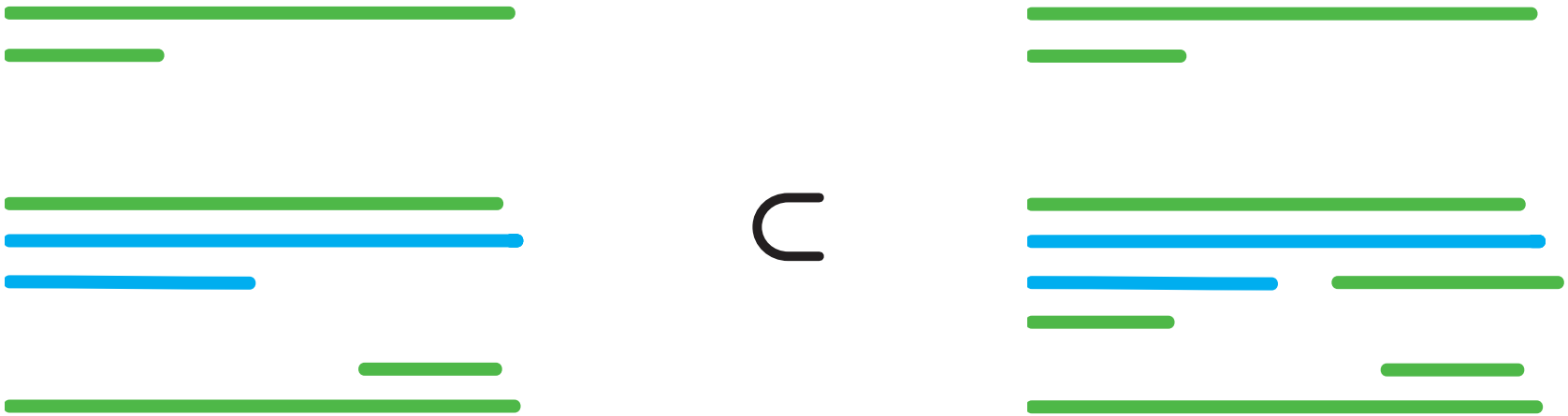
- We extract sentences (green) as a summary of the full document



- The summary on the left is a subset of the summary on the right.
- add new (blue) sentence to each of the two summaries.

Extractive Document Summarization

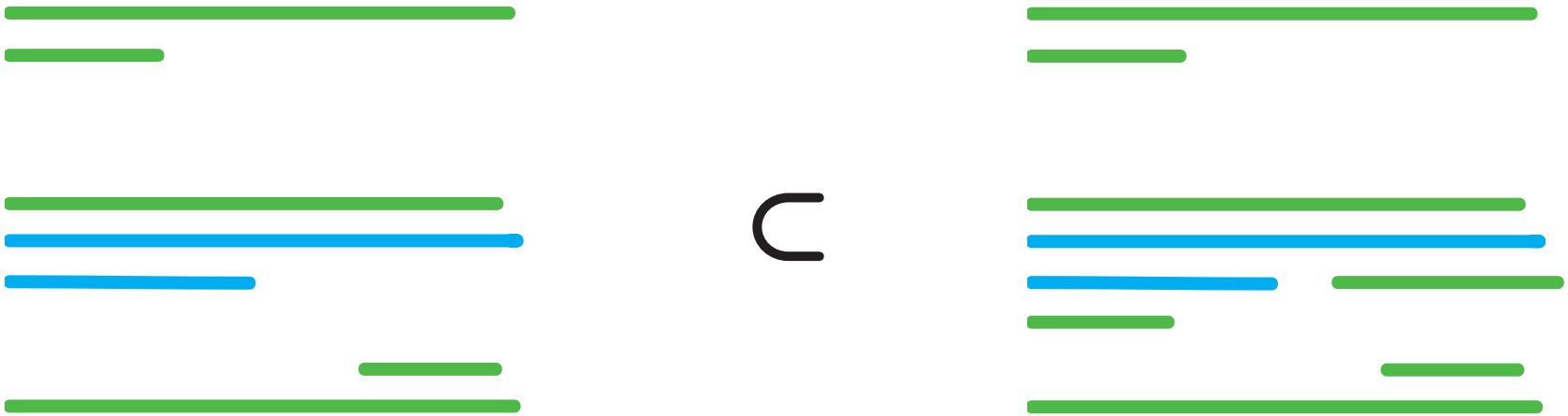
- We extract sentences (green) as a summary of the full document



- The summary on the left is a subset of the summary on the right.
- add new (blue) sentence to each of the two summaries.
- The marginal (incremental) benefit of adding the new (blue) sentence to the smaller (left) summary is no more than the marginal benefit of adding the new sentence to the larger (right) summary.

Extractive Document Summarization

- We extract sentences (green) as a summary of the full document



- The summary on the left is a subset of the summary on the right.
- add new (blue) sentence to each of the two summaries.
- The marginal (incremental) benefit of adding the new (blue) sentence to the smaller (left) summary is no more than the marginal benefit of adding the new sentence to the larger (right) summary.
- **diminishing returns** $\leftrightarrow$ **submodularity**

Problem setup

- The ground set V corresponds to all the sentences in a document.
- Extractive document summarization: select a small subset $S \subseteq V$ that accurately represents the entirety (ground set V).
- The summary is usually required to be length-limited.
  - c_i: cost (e.g., the number of words in sentence i),
  - b: the budget (e.g., the largest length allowed),
  - knapsack constraint: $\sum_{i \in S} c_i \leq b$.
- A set function $f : 2^V \rightarrow \mathbb{R}$ measures the quality of the summary S,
- Thus, the summarization problem is formalized as:

Problem (Document Summarization Optimization Problem)

$$S^* \in \operatorname*{argmax}_{S \subseteq V} f(S) \text{ subject to: } \sum_{i \in S} c_i \leq b. \quad (1)$$

A Practical Algorithm for Large-Scale Summarization

When f is both **monotone** and **submodular**:

- A greedy algorithm with partial enumeration (Sviridenko, 2004), theoretical guarantee of near-optimal solution, but not practical for large data sets, $O(n^3)$.

A Practical Algorithm for Large-Scale Summarization

When f is both **monotone** and **submodular**:

- A greedy algorithm with partial enumeration (Sviridenko, 2004), theoretical guarantee of near-optimal solution, but not practical for large data sets, $O(n^3)$.
- A greedy algorithm (Lin and Bilmes, 2010): with worse theoretical guarantee but still constant factor $1 - 1/\sqrt{e} \approx 0.39$, and practical/scalable (e.g., Minoux trick still works)!
  - We choose next element with largest ratio of gain over scaled cost:

$$k \leftarrow \operatorname*{argmax}_{i \in U} \frac{f(G \cup \{i\}) - f(G)}{(c_i)^r}. \tag{2}$$

A Practical Algorithm for Large-Scale Summarization

When f is both **monotone** and **submodular**:

- A greedy algorithm with partial enumeration (Sviridenko, 2004), theoretical guarantee of near-optimal solution, but not practical for large data sets, $O(n^3)$.
- A greedy algorithm (Lin and Bilmes, 2010): with worse theoretical guarantee but still constant factor $1 - 1/\sqrt{e} \approx 0.39$, and practical/scalable (e.g., Minoux trick still works)!
  - We choose next element with largest ratio of gain over scaled cost:

$$k \leftarrow \operatorname*{argmax}_{i \in U} \frac{f(G \cup \{i\}) - f(G)}{(c_i)^r}. \tag{2}$$

  - Scalability: the argmax above can typically be solved by $O(\log n)$ calls of f, thanks to submodularity
  - Ex: integer linear programming (ILP) takes 17 hours vs. greedy which takes < 1 second!!

Outline

The General Form of Our Submodular Functions

- Two properties of a good summary: relevance and non-redundancy.
- A redundancy penalty often violates monotonicity.
- Our approach: we positively **reward diversity** instead of negatively penalizing redundancy:

Definition (The general form of our submodular functions)

$$f(S) = \mathcal{L}(S) + \lambda \mathcal{R}(S)$$

- $\mathcal{L}(S)$ measures the coverage (or fidelity) of summary set S to the document.
- $\mathcal{R}(S)$ rewards diversity in S.
- $\lambda \geq 0$ is a trade-off coefficient.

Coverage function

Coverage Function

$$\mathcal{L}(S) = \sum_{i \in V} \min \{\mathcal{C}_i(S), \alpha\, \mathcal{C}_i(V)\}$$

- $\mathcal{C}_i : 2^V \rightarrow \mathbb{R}$ is monotone submodular, and measures how well i is covered by S: $\Rightarrow \mathcal{L}(S)$ is monotone submodular.
- $0 \leq \alpha \leq 1$ is a threshold coefficient — sufficient coverage fraction.
- if $\min\{\mathcal{C}_i(S), \alpha\mathcal{C}_i(V)\} = \alpha\mathcal{C}_i(V)$, then sentence i is well covered by summary S (saturated).
- After saturation, further increases in $\mathcal{C}_i(S)$ won't increase the objective function values (return diminishes).
- Therefore, new sentence added to S should focus on sentences that are not yet saturated, in order to increasing the objective function value.

Coverage function

Coverage Function

$$\mathcal{L}(S) = \sum_{i \in V} \min \{\mathcal{C}_i(S), \alpha\, \mathcal{C}_i(V)\}$$

- $\mathcal{C}_i$ measures how well i is covered by S.
- One simple possible $\mathcal{C}_i$ (that we use) is:

$$\mathcal{C}_i(S) = \sum_{j \in S} w_{i,j},$$

  where $w_{i,j} \geq 0$ measures the similarity between i and j.
- With this C_i, $\mathcal{L}(S)$ is monotone submodular, as required.

Diversity reward function

Diversity Reward Function

$$\mathcal{R}(S) = \sum_{i=1}^{K} \sqrt{\sum_{j \in P_i \cap S} r_j}.$$

- $\mathcal{P} = \{P_i : i = 1, \cdots K\}$ is a partition of the ground set V
- $r_j \geq 0$: **singleton reward** of j, which represents the importance of j to the summary.
- square root over the sum of rewards of sentences belong to the same partition (diminishing returns).
- $\mathcal{R}(S)$ is monotone submodular as well.

Diversity Reward Function

Alternatively, we can utilize multiple partitions/clusterings $\mathcal{P}_1, \mathcal{P}_2, \ldots,$ yielding diversity reward function for each one, and mix them together.

Multi-resolution Diversity Reward

$$\mathcal{R}(S) = \lambda_1 \sum_{i=1}^{K_1} \sqrt{\sum_{j \in P_i^{(1)} \cap S} r_j} + \lambda_2 \sum_{i=1}^{K_2} \sqrt{\sum_{j \in P_i^{(2)} \cap S} r_j} + \cdots$$

Outline

DUC Evaluation

- There are standard evaluation data sets for document summarization.

DUC Evaluation

- There are standard evaluation data sets for document summarization.
- DUC (Document Understanding Conference) was run by NIST `http://duc.nist.gov/` from 2003-2007.

DUC Evaluation

- There are standard evaluation data sets for document summarization.
- DUC (Document Understanding Conference) was run by NIST `http://duc.nist.gov/` from 2003-2007.
- Researchers have continued to use these data sets since they are a widely-used standard and $\exists$ automatic evaluation strategy (ROUGE).

DUC Evaluation

- There are standard evaluation data sets for document summarization.
- DUC (Document Understanding Conference) was run by NIST `http://duc.nist.gov/` from 2003-2007.
- Researchers have continued to use these data sets since they are a widely-used standard and $\exists$ automatic evaluation strategy (ROUGE).
- Generic summarization (not in response to a query): DUC 2004

DUC Evaluation

- There are standard evaluation data sets for document summarization.
- DUC (Document Understanding Conference) was run by NIST `http://duc.nist.gov/` from 2003-2007.
- Researchers have continued to use these data sets since they are a widely-used standard and $\exists$ automatic evaluation strategy (ROUGE).
- Generic summarization (not in response to a query): DUC 2004
- Query-based summarization (user issues a query and summary must be relevant to query): DUC 2005-2007 (more like web search/IR).

Generic Summarization

- DUC-04: generic summarization

Table : ROUGE-1 recall (R) and F-measure (F) results (%) on DUC-04. DUC-03 was used as development set.

DUC-04	R	F
$\mathcal{L}_1(S)$	39.03	38.65
$\mathcal{R}_1(S)$	38.23	37.81
$\mathcal{L}_1(S) + \lambda\mathcal{R}_1(S)$	**39.35**	**38.90**
Takamura and Okumura (2009)	38.50	-
Wang et al. (2009)	39.07	-
Lin and Bilmes (2010)	-	38.39
Best system in DUC-04 (peer 65)	38.28	37.94

- Note: this was (in 2011) the best ROUGE-1 result ever reported on DUC-04.

Query-focused Summarization

- DUC-05,06,07: query-focused summarization
- For each document cluster, a title and a narrative (query) describing a user's information need are provided.
- Nelder-Mead (derivative-free) for parameter training.

DUC-05 results

Table : ROUGE-2 recall (R) and F-measure (F) results (%)

	R	F
$\mathcal{L}_1(S) + \lambda \mathcal{R}_Q(S)$	7.82	7.72
$\mathcal{L}_1(S) + \sum_{\kappa=1}^{3} \lambda_\kappa \mathcal{R}_{Q,\kappa}(S)$	**8.19**	**8.13**
Daumé III and Marcu (2006)	6.98	-
Wei et al. (2010)	8.02	-
Best system in DUC-05 (peer 15)	7.44	7.43

- DUC-06 was used as training set for the objective function with single diversity reward.
- DUC-06 and 07 were used as training sets for the objective function with multi-resolution diversity reward
- Note: this was (in 2011) the best ROUGE-2 result ever reported on DUC-05.

DUC-06 results

Table : ROUGE-2 recall (R) and F-measure (F) results (%)

	R	F
$\mathcal{L}_1(S) + \lambda \mathcal{R}_Q(S)$	9.75	9.77
$\mathcal{L}_1(S) + \sum_{\kappa=1}^{3} \lambda_\kappa \mathcal{R}_{Q,\kappa}(S)$	**9.81**	**9.82**
Celikyilmaz and Hakkani-tür (2010)	9.10	-
Shen and Li (2010)	9.30	-
Best system in DUC-06 (peer 24)	9.51	9.51

- DUC-05 was used as training set for the objective function with single diversity reward.
- DUC-05 and 07 were used as training sets for the objective function with multi-resolution diversity reward
- Note: this was (in 2011) the best ROUGE-2 result ever reported on DUC-06.

DUC-07 results

Table : ROUGE-2 recall (R) and F-measure (F) results (%)

	R	F
$\mathcal{L}_1(S) + \lambda \mathcal{R}_Q(S)$	12.18	12.13
$\mathcal{L}_1(S) + \sum_{\kappa=1}^{3} \lambda_\kappa \mathcal{R}_{Q,\kappa}(S)$	**12.38**	**12.33**
Toutanova et al. (2007)	11.89	11.89
Haghighi and Vanderwende (2009)	11.80	-
Celikyilmaz and Hakkani-tür (2010)	11.40	-
Best system in DUC-07 (peer 15), using web search	**12.45**	12.29

- DUC-05 was used as training set for the objective function with single diversity reward.
- DUC-05 and 06 were used as training sets for the objective function with multi-resolution diversity reward.
- Note: this was (in 2011) the best ROUGE-2 F-measure result ever reported on DUC-07, and best ROUGE-2 R without web search expansion.

Outline

Learning Submodular Functions

- The ROUGE-N score is submodular but is inaccessible.

Learning Submodular Functions

- The ROUGE-N score is submodular but is inaccessible.
- Above: we have deduced by hand a class of submodular function as a surrogate to ROUGE.

Learning Submodular Functions

- The ROUGE-N score is submodular but is inaccessible.
- Above: we have deduced by hand a class of submodular function as a surrogate to ROUGE.
- Can we directly learn ROUGE function given supervised data?

Learning Submodular Functions

- The ROUGE-N score is submodular but is inaccessible.
- Above: we have deduced by hand a class of submodular function as a surrogate to ROUGE.
- Can we directly learn ROUGE function given supervised data?
- *Goemans et al. (2009)*: "can one make only polynomial number of queries to an unknown submodular function f and constructs a $\hat{f}$ such that $\hat{f}(S) \leq f(S) \leq g(n)\hat{f}(S)$ where $g : \mathbb{N} \rightarrow \mathbb{R}$?"

Learning Submodular Functions

- The ROUGE-N score is submodular but is inaccessible.
- Above: we have deduced by hand a class of submodular function as a surrogate to ROUGE.
- Can we directly learn ROUGE function given supervised data?
- *Goemans et al. (2009)*: "can one make only polynomial number of queries to an unknown submodular function f and constructs a $\hat{f}$ such that $\hat{f}(S) \leq f(S) \leq g(n)\hat{f}(S)$ where $g : \mathbb{N} \rightarrow \mathbb{R}$?" Many results, including that even with adaptive queries and monotone functions, can't do better than $\Omega(\sqrt{n}/\log n)$.

Learning Submodular Functions

- The ROUGE-N score is submodular but is inaccessible.
- Above: we have deduced by hand a class of submodular function as a surrogate to ROUGE.
- Can we directly learn ROUGE function given supervised data?
- *Goemans et al. (2009)*: "can one make only polynomial number of queries to an unknown submodular function f and constructs a $\hat{f}$ such that $\hat{f}(S) \leq f(S) \leq g(n)\hat{f}(S)$ where $g : \mathbb{N} \rightarrow \mathbb{R}$?" Many results, including that even with adaptive queries and monotone functions, can't do better than $\Omega(\sqrt{n}/\log n)$.
- *Balcan & Harvey (2011)*: submodular function learning problem from a learning theory perspective, given a distribution on subsets. Negative result is that can't approximate in this setting to within a constant factor.

Learning Submodular Functions

- The ROUGE-N score is submodular but is inaccessible.
- Above: we have deduced by hand a class of submodular function as a surrogate to ROUGE.
- Can we directly learn ROUGE function given supervised data?
- *Goemans et al. (2009)*: "can one make only polynomial number of queries to an unknown submodular function f and constructs a $\hat{f}$ such that $\hat{f}(S) \leq f(S) \leq g(n)\hat{f}(S)$ where $g : \mathbb{N} \rightarrow \mathbb{R}$?" Many results, including that even with adaptive queries and monotone functions, can't do better than $\Omega(\sqrt{n}/\log n)$.
- *Balcan & Harvey (2011)*: submodular function learning problem from a learning theory perspective, given a distribution on subsets. Negative result is that can't approximate in this setting to within a constant factor.
- Learning submodular functions is hard!

Submodular Mixtures

- Learning submodular functions with unknown forms/structures is hard.

Submodular Mixtures

- Learning submodular functions with unknown forms/structures is hard.
- But this does not preclude learning submodular functions with known forms with unknown parameters.

Submodular Mixtures

- Learning submodular functions with unknown forms/structures is hard.
- But this does not preclude learning submodular functions with known forms with unknown parameters.
- Function class: conic combination of a finite number of submodular functions with known forms.

Submodular Mixtures

- Learning submodular functions with unknown forms/structures is hard.
- But this does not preclude learning submodular functions with known forms with unknown parameters.
- Function class: conic combination of a finite number of submodular functions with known forms.
- $\Rightarrow$ submodular mixtures, where each submodular function is a component of the mixture.

Submodular Mixtures

- Learning submodular functions with unknown forms/structures is hard.
- But this does not preclude learning submodular functions with known forms with unknown parameters.
- Function class: conic combination of a finite number of submodular functions with known forms.
- $\Rightarrow$ submodular mixtures, where each submodular function is a component of the mixture.
- A fairly rich class of monotone submodular functions can be represented as a submodular mixture of components with simple forms.

Structured Prediction in Machine Learning

- Given: a finite set of training pairs $D = \left\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\right\}_i$ where $\mathbf{x}^{(i)} \in \mathcal{X}$, $\mathbf{y}^{(i)} \in \mathcal{Y}$.

Structured Prediction in Machine Learning

- Given: a finite set of training pairs $D = \left\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\right\}_i$ where $\mathbf{x}^{(i)} \in \mathcal{X}$, $\mathbf{y}^{(i)} \in \mathcal{Y}$.
- $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^M$ is a (fixed) vector of functions, and $\mathbf{w} \in \mathbb{R}^M$ is a vector.

Structured Prediction in Machine Learning

- Given: a finite set of training pairs $D = \left\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\right\}_i$ where $\mathbf{x}^{(i)} \in \mathcal{X}$, $\mathbf{y}^{(i)} \in \mathcal{Y}$.
- $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^M$ is a (fixed) vector of functions, and $\mathbf{w} \in \mathbb{R}^M$ is a vector.
- Score function: $s(\mathbf{x}, \mathbf{y}) = \mathbf{w}^\intercal \mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_i w_i f_i(\mathbf{x}, \mathbf{y})$.

Structured Prediction in Machine Learning

- Given: a finite set of training pairs $D = \left\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\right\}_i$ where $\mathbf{x}^{(i)} \in \mathcal{X}$, $\mathbf{y}^{(i)} \in \mathcal{Y}$.
- $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^M$ is a (fixed) vector of functions, and $\mathbf{w} \in \mathbb{R}^M$ is a vector.
- Score function: $s(\mathbf{x}, \mathbf{y}) = \mathbf{w}^\intercal \mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_i w_i f_i(\mathbf{x}, \mathbf{y})$.
- Decision making (inference) for a given $\bar{\mathbf{x}}$ is based on:

$$\hat{\mathbf{y}} \in h_{\mathbf{w}}(\bar{\mathbf{x}}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} s(\bar{\mathbf{x}}, \mathbf{y}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\intercal \mathbf{f}(\bar{\mathbf{x}}, \mathbf{y}) \tag{3}$$

Structured Prediction in Machine Learning

- Given: a finite set of training pairs $D = \left\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\right\}_i$ where $\mathbf{x}^{(i)} \in \mathcal{X}$, $\mathbf{y}^{(i)} \in \mathcal{Y}$.
- $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^M$ is a (fixed) vector of functions, and $\mathbf{w} \in \mathbb{R}^M$ is a vector.
- Score function: $s(\mathbf{x}, \mathbf{y}) = \mathbf{w}^\intercal \mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_i w_i f_i(\mathbf{x}, \mathbf{y})$.
- Decision making (inference) for a given $\bar{\mathbf{x}}$ is based on:

$$\hat{\mathbf{y}} \in h_{\mathbf{w}}(\bar{\mathbf{x}}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} s(\bar{\mathbf{x}}, \mathbf{y}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\intercal \mathbf{f}(\bar{\mathbf{x}}, \mathbf{y}) \tag{3}$$

- Goal of "learning" is to optimize $\mathbf{w}$ so that such decision making is "good"

Structured Prediction in Machine Learning

- Given: a finite set of training pairs $D = \left\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\right\}_i$ where $\mathbf{x}^{(i)} \in \mathcal{X}$, $\mathbf{y}^{(i)} \in \mathcal{Y}$.
- $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^M$ is a (fixed) vector of functions, and $\mathbf{w} \in \mathbb{R}^M$ is a vector.
- Score function: $s(\mathbf{x}, \mathbf{y}) = \mathbf{w}^\intercal \mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_i w_i f_i(\mathbf{x}, \mathbf{y})$.
- Decision making (inference) for a given $\bar{\mathbf{x}}$ is based on:

$$\hat{\mathbf{y}} \in h_{\mathbf{w}}(\bar{\mathbf{x}}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} s(\bar{\mathbf{x}}, \mathbf{y}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\intercal \mathbf{f}(\bar{\mathbf{x}}, \mathbf{y}) \tag{3}$$

- Goal of "learning" is to optimize $\mathbf{w}$ so that such decision making is "good"
- Let $\ell : \mathcal{Y} \times Y \to \mathbb{R}_+$ be a loss function. I.e., $\ell_{\mathbf{y}}(\hat{\mathbf{y}})$ is cost of deciding $\hat{\mathbf{y}}$ when truth is $\mathbf{y}$.

Structured Prediction in Machine Learning

- Given: a finite set of training pairs $D = \left\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\right\}_i$ where $\mathbf{x}^{(i)} \in \mathcal{X}$, $\mathbf{y}^{(i)} \in \mathcal{Y}$.
- $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^M$ is a (fixed) vector of functions, and $\mathbf{w} \in \mathbb{R}^M$ is a vector.
- Score function: $s(\mathbf{x}, \mathbf{y}) = \mathbf{w}^\intercal \mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_i w_i f_i(\mathbf{x}, \mathbf{y})$.
- Decision making (inference) for a given $\bar{\mathbf{x}}$ is based on:

$$\hat{\mathbf{y}} \in h_{\mathbf{w}}(\bar{\mathbf{x}}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} s(\bar{\mathbf{x}}, \mathbf{y}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\intercal \mathbf{f}(\bar{\mathbf{x}}, \mathbf{y}) \tag{3}$$

- Goal of "learning" is to optimize $\mathbf{w}$ so that such decision making is "good"
- Let $\ell : \mathcal{Y} \times Y \to \mathbb{R}_+$ be a loss function. I.e., $\ell_{\mathbf{y}}(\hat{\mathbf{y}})$ is cost of deciding $\hat{\mathbf{y}}$ when truth is $\mathbf{y}$.
- Empirical risk minimization: adjust $\mathbf{w}$ so that $\sum_i \ell_{\mathbf{y}}(h_{\mathbf{w}}(\mathbf{x}^{(i)}))$ is small subject to other conditions (e.g., regularization).

Structured Prediction: Approach

- Minimization via convex programming

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{4}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) \geq \ell_t(\mathbf{y}) - \xi_t, \forall t, \forall \mathbf{y} \in \mathcal{Y}_t \tag{5}$$

$$\xi_t \geq 0, \forall t. \tag{6}$$

where $\mathbf{f}_t(\mathbf{y}) = \mathbf{f}(\mathbf{x}_t, \mathbf{y})$, $\ell_t(\mathbf{y}) = \ell_{\mathbf{y}_t}(\mathbf{y})$,

Structured Prediction: Approach

- Minimization via convex programming

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{4}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) \geq \ell_t(\mathbf{y}) - \xi_t, \forall t, \forall \mathbf{y} \in \mathcal{Y}_t \tag{5}$$

$$\xi_t \geq 0, \forall t. \tag{6}$$

where $\mathbf{f}_t(\mathbf{y}) = \mathbf{f}(\mathbf{x}_t, \mathbf{y})$, $\ell_t(\mathbf{y}) = \ell_{\mathbf{y}_t}(\mathbf{y})$,

- Exponential number of constraints, due to the $\forall \mathbf{y} \in \mathcal{Y}_t$

Structured Prediction: Approach with inference

- Constraints specified in inference form:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{7}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t} \Big(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\Big) - \xi_t, \forall t \tag{8}$$

$$\xi_t \geq 0, \forall t. \tag{9}$$

Structured Prediction: Approach with inference

- Constraints specified in inference form:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{7}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t} \Big(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\Big) - \xi_t, \forall t \tag{8}$$

$$\xi_t \geq 0, \forall t. \tag{9}$$

- Exponential set of constraints reduced to an embedded optimization problem, "inference."

Structured Prediction: Unconstrained form

- Unconstrained form with hinge-loss

$$\min_{\mathbf{w}} \frac{1}{T} \sum_t \left[\max_{\mathbf{y} \in \mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \right] + \frac{\lambda}{2} \|\mathbf{w}\|^2 \quad (10)$$

Structured Prediction: Unconstrained form

- Unconstrained form with hinge-loss

$$\min_{\mathbf{w}} \frac{1}{T} \sum_t \left[\max_{\mathbf{y} \in \mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \right] + \frac{\lambda}{2} \|\mathbf{w}\|^2 \quad (10)$$

- This generalizes the hinge-loss function with loss as follows:

$$\ell^{\text{hinge}}_{\mathbf{y}^{(t)}}(h(\mathbf{x})) = \max_{\mathbf{y} \in \mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \quad (11)$$

Structured Prediction: Unconstrained form

- Unconstrained form with hinge-loss

$$\min_{\mathbf{w}} \frac{1}{T} \sum_t \left[\max_{\mathbf{y} \in \mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \right] + \frac{\lambda}{2} \|\mathbf{w}\|^2 \quad (10)$$

- This generalizes the hinge-loss function with loss as follows:

$$\ell^{\text{hinge}}_{\mathbf{y}^{(t)}}(h(\mathbf{x})) = \max_{\mathbf{y} \in \mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \quad (11)$$

- $\ell^{\text{hinge}}_{\mathbf{y}}(h(\mathbf{x}))$ convex in $\mathbf{w}$.

Structured Prediction: Subgradient

- Subgradient, evaluated at $\mathbf{w}$, of the following

$$\max_{\mathbf{y} \in \mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \qquad (12)$$

Structured Prediction: Subgradient

- Subgradient, evaluated at $\mathbf{w}$, of the following

$$\max_{\mathbf{y} \in \mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \tag{12}$$

can be found by computing

$$\mathbf{y}^* \in \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \tag{13}$$

Structured Prediction: Subgradient

- Subgradient, evaluated at $\mathbf{w}$, of the following

$$\max_{\mathbf{y} \in \mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \tag{12}$$

can be found by computing

$$\mathbf{y}^* \in \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \tag{13}$$

and then finding subgradient of

$$\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^*) + \ell_t(\mathbf{y}^*) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \tag{14}$$

Structured Prediction: Subgradient

- Subgradient, evaluated at $\mathbf{w}$, of the following

$$\max_{\mathbf{y} \in \mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \tag{12}$$

can be found by computing

$$\mathbf{y}^* \in \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \tag{13}$$

and then finding subgradient of

$$\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^*) + \ell_t(\mathbf{y}^*) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \tag{14}$$

which has the form

$$\mathbf{f}_t(\mathbf{y}^*) - \mathbf{f}_t(\mathbf{y}^{(t)}) + \lambda \mathbf{w}. \tag{15}$$

Structured Prediction: Subgradient Learning

Algorithm 1: Subgradient descent learning

Input : $S = \{(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})\}_{t=1}^{T}$ and a learning rate sequence $\{\eta_t\}_{t=1}^{T}$.

$w_0 = 0$;

for $t = 1, \cdots, T$ **do**

- Loss augmented inference: $\mathbf{y}_t^* \in \text{argmax}_{\mathbf{y} \in \mathcal{Y}_t} \mathbf{w}_{t-1}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})$;
- Compute the subgradient: $\mathbf{g}_t = \lambda \mathbf{w}_{t-1} + \mathbf{f}_t(\mathbf{y}^*) - \mathbf{f}_t(\mathbf{y}^{(t)})$;
- Update the weights: $\mathbf{w}_t = \mathbf{w}_{t-1} - \eta_t \mathbf{g}_t$;

Return : the averaged parameters $\frac{1}{T} \sum_t \mathbf{w}_t$.

Structured Prediction: Approximate Inference

- Assumption has been $\mathbf{y}_t^* \in \text{argmax}_{\mathbf{y} \in \mathcal{Y}_t} \mathbf{w}_{t-1}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})$ is exact.

Structured Prediction: Approximate Inference

- Assumption has been $\mathbf{y}_t^* \in \text{argmax}_{\mathbf{y} \in \mathcal{Y}_t} \mathbf{w}_{t-1}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})$ is exact.
- Past work has often required assumptions on $\mathbf{f}$ (e.g., decomposability) and/or ℓ (e.g., Hamming loss) that makes the inference tractable.

Structured Prediction: Approximate Inference

- Assumption has been $\mathbf{y}_t^* \in \text{argmax}_{\mathbf{y} \in \mathcal{Y}_t} \mathbf{w}_{t-1}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})$ is exact.
- Past work has often required assumptions on $\mathbf{f}$ (e.g., decomposability) and/or ℓ (e.g., Hamming loss) that makes the inference tractable.
- If inference is approximate, the learning (and risk bounds) are not always guaranteed to exist (Kulesza & Pereira 2007) (loopy-belief propagation based inference dissolves the guarantee for perceptron learning).

Structured Prediction: Approximate Inference

- Assumption has been $\mathbf{y}_t^* \in \text{argmax}_{\mathbf{y} \in \mathcal{Y}_t} \mathbf{w}_{t-1}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})$ is exact.
- Past work has often required assumptions on $\mathbf{f}$ (e.g., decomposability) and/or ℓ (e.g., Hamming loss) that makes the inference tractable.
- If inference is approximate, the learning (and risk bounds) are not always guaranteed to exist (Kulesza & Pereira 2007) (loopy-belief propagation based inference dissolves the guarantee for perceptron learning).
- Fortunately, we show in our case (submodular maximization), that such bounds do exist.

Submodular Mixture Score Functions

Inference (decoding) problem in NLP:

- Input: $\mathbf{x} \in \mathcal{X}$
- Output: $\mathbf{y} \in \mathcal{Y}$
- Score function: $s : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$
- Inference: $\text{argmax}_{\mathbf{y} \in \mathcal{Y}} s(\mathbf{x}, \mathbf{y})$
- Submodular mixture score:

$$s(x, y) = \sum_i w_i f_i(x, y) \tag{16}$$

  where $w \geq 0$ and $\forall i, f_i : \mathcal{X} \times \mathcal{Y}_{\mathbf{x}} \to \mathbb{R}$ is *submodular* on $\mathcal{Y}_{\mathbf{x}}$.
- f_i: component of the mixture, which is given.
- w_i: component weight, which is unknown
- **Goal**: supervised learning of component weights $\mathbf{w}$.

Submodular Components

Many possible components

- Weighted sums of weighted matroid rank functions (Shioura, 2012)
  Given matroids $\mathcal{M}_i = (V, \mathcal{I}_i)$, and modular functions $m_i : 2^V \rightarrow \mathbb{R}_+$, class is:

$$f(S) = \sum_{i=1}^{M} w_i \max \{m_i(I) : I \subseteq S, I \in \mathcal{I}_i\} \tag{17}$$

  can easily cover cover-like functions,

Submodular Components

Many possible components

- Weighted sums of weighted matroid rank functions (Shioura, 2012)
  Given matroids $\mathcal{M}_i = (V, \mathcal{I}_i)$, and modular functions $m_i : 2^V \rightarrow \mathbb{R}_+$, class is:

$$f(S) = \sum_{i=1}^{M} w_i \max \{m_i(I) : I \subseteq S, I \in \mathcal{I}_i\} \tag{17}$$

  can easily cover cover-like functions, a broader class than $M^\natural$-concave functions.

Submodular Components

Many possible components

- Weighted sums of weighted matroid rank functions (Shioura, 2012)
  Given matroids $\mathcal{M}_i = (V, \mathcal{I}_i)$, and modular functions $m_i : 2^V \rightarrow \mathbb{R}_+$, class is:

$$f(S) = \sum_{i=1}^{M} w_i \max\{m_i(I) : I \subseteq S, I \in \mathcal{I}_i\} \tag{17}$$

  can easily cover cover-like functions, a broader class than $M^\natural$-concave functions.
- Truncation functions of the form $f(S) = \min(m(S), \alpha)$ for modular m.

Submodular Components

Many possible components

- Weighted sums of weighted matroid rank functions (Shioura, 2012) Given matroids $\mathcal{M}_i = (V, \mathcal{I}_i)$, and modular functions $m_i : 2^V \rightarrow \mathbb{R}_+$, class is:

$$f(S) = \sum_{i=1}^{M} w_i \max\{m_i(I) : I \subseteq S, I \in \mathcal{I}_i\} \tag{17}$$

  can easily cover cover-like functions, a broader class than $M^\natural$-concave functions.
- Truncation functions of the form $f(S) = \min(m(S), \alpha)$ for modular m.
- Or any polymatroid functions, such as the ones we previously used in the fixed mixtures for summarization.

Submodular Components

Many possible components

- Weighted sums of weighted matroid rank functions (Shioura, 2012) Given matroids $\mathcal{M}_i = (V, \mathcal{I}_i)$, and modular functions $m_i : 2^V \rightarrow \mathbb{R}_+$, class is:

$$f(S) = \sum_{i=1}^{M} w_i \max\{m_i(I) : I \subseteq S, I \in \mathcal{I}_i\} \tag{17}$$

  can easily cover cover-like functions, a broader class than $M^\natural$-concave functions.
- Truncation functions of the form $f(S) = \min(m(S), \alpha)$ for modular m.
- Or any polymatroid functions, such as the ones we previously used in the fixed mixtures for summarization.
- Key is that they need to be specified beforehand.

Submodular Components

Many possible components

- Weighted sums of weighted matroid rank functions (Shioura, 2012)
  Given matroids $\mathcal{M}_i = (V, \mathcal{I}_i)$, and modular functions $m_i : 2^V \rightarrow \mathbb{R}_+$, class is:

$$f(S) = \sum_{i=1}^{M} w_i \max\{m_i(I) : I \subseteq S, I \in \mathcal{I}_i\} \tag{17}$$

  can easily cover cover-like functions, a broader class than $M^\natural$-concave functions.
- Truncation functions of the form $f(S) = \min(m(S), \alpha)$ for modular m.
- Or any polymatroid functions, such as the ones we previously used in the fixed mixtures for summarization.
- Key is that they need to be specified beforehand.
- Total number of available components M must be fixed and finite.

Learning Submodular Mixtures: Unconstrained Form

- Unconstrained form with hinge-loss

$$\min_{\mathbf{w}\geq 0} \frac{1}{T} \sum_t \left[\max_{\mathbf{y}\in\mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \right] + \frac{\lambda}{2} \|\mathbf{w}\|^2 \quad (18)$$

Learning Submodular Mixtures: Unconstrained Form

- Unconstrained form with hinge-loss

$$\min_{\mathbf{w}\geq 0} \frac{1}{T} \sum_t \left[\max_{\mathbf{y}\in\mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \right] + \frac{\lambda}{2} \|\mathbf{w}\|^2 \quad (18)$$

- To compute a subgradient, we must solve the following embedded optimization problem

$$\max_{\mathbf{y}\in\mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) \quad (19)$$

Learning Submodular Mixtures: Unconstrained Form

- Unconstrained form with hinge-loss

$$\min_{\mathbf{w}\geq 0} \frac{1}{T} \sum_t \left[\max_{\mathbf{y}\in\mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \right] + \frac{\lambda}{2} \|\mathbf{w}\|^2 \quad (18)$$

- To compute a subgradient, we must solve the following embedded optimization problem

$$\max_{\mathbf{y}\in\mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) \quad (19)$$

- Convex in $\mathbf{w}$, and $\mathbf{w}^\top \mathbf{f}_t(\mathbf{y})$ presumably polymatroidal, but what about $\ell_t(\mathbf{y})$?

Learning Submodular Mixtures: Unconstrained Form

- Unconstrained form with hinge-loss

$$\min_{\mathbf{w}\geq 0} \frac{1}{T} \sum_t \left[\max_{\mathbf{y}\in\mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \right] + \frac{\lambda}{2} \|\mathbf{w}\|^2 \quad (18)$$

- To compute a subgradient, we must solve the following embedded optimization problem

$$\max_{\mathbf{y}\in\mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) \quad (19)$$

- Convex in $\mathbf{w}$, and $\mathbf{w}^\top \mathbf{f}_t(\mathbf{y})$ presumably polymatroidal, but what about $\ell_t(\mathbf{y})$?
- Often one uses Hamming loss, but here lets assume that $\ell_t(\mathbf{y})$ is at least polymatroidal (more soon on this).

Learning Submodular Mixtures

Algorithm 2: Projected subgradient descent for learning submodular mixtures.

Input : $S = \{(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})\}_{t=1}^{T}$ and a learning rate sequence $\{\eta_t\}_{t=1}^{T}$.

1 $w_0 = 0$;

2 **for** $t = 1, \cdots, T$ **do**

3 Approximate inference: $\hat{\mathbf{y}} \in \text{arg}\approx\max_{\mathbf{y}\in\mathcal{Y}_t} \mathbf{w}_{t-1}^{\top}\mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})$;

4 Compute the subgradient: $\mathbf{g}_t = \lambda\mathbf{w}_{t-1} + \mathbf{f}_t(\hat{\mathbf{y}}) - \mathbf{f}_t(\mathbf{y}^{(t)})$;

5 Update the weights with projection: $\mathbf{w}_t = \max(\mathbf{0}, \mathbf{w}_{t-1} - \eta_t\mathbf{g}_t)$;

Return : the averaged parameters $\frac{1}{T}\sum_t \mathbf{w}_t$.

- Note line 3, the approximate maximization (e.g., submodular maximization).
- Note line 5, the projection step, to ensure $\mathbf{w} \geq 0$ which preserves submodularity.

Learning submodular mixtures with subgradient decent

Theorem (Lin & Bilmes 2012)

Assume $f_i, i = 1, \cdots, M$ *are all upper-bounded by 1,* $r_t(\mathbf{w}) \leq B$, *and* $\|\mathbf{g}_t\| \leq G$. *Let* $\hat{\mathbf{w}}$ *be the solution returned by Algorithm 2 using* ρ*-approximate inference with learning rate* $\eta_t = \frac{2}{\lambda t}$ *and* $\lambda = \frac{G}{M}\sqrt{\frac{2(1+\log T)}{T}}$. *Then for any* $\delta > 0$ *with probability at least* $1 - \delta$,

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D}[\ell_{\mathbf{y}}(h(\mathbf{x};\hat{\mathbf{w}}))] \leq \frac{1}{\rho}\left(\frac{1}{T}\sum_{t=1}^{T} r_t(\mathbf{w}^*)\right) + S(T),$$

where

$$S(T) = \frac{MG}{\rho}\sqrt{\frac{2(1+\log T)}{T}} + B\sqrt{\frac{2}{T}\log\frac{1}{\delta}} + \frac{1-\rho}{\rho}M$$

and

$$r_t(\mathbf{w}) = \max_{\mathbf{y}\in\mathcal{Y}_t}\left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\right) - \mathbf{w}^\mathsf{T}\mathbf{f}_t(\mathbf{y}^{(t)}) \tag{20}$$

Risk bound, learning submodular mixtures with subgradient decent

Theorem (Lin & Bilmes 2012 (summarized))

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D}[\ell(h(\mathbf{x};\hat{\mathbf{w}}),\mathbf{y})] \leq \frac{1}{\rho}\left(\frac{1}{n}\sum_{t=1}^{n} r_t(\mathbf{w}^*)\right) + S(n),$$

where

$$S(n) = \frac{MG}{\rho}\sqrt{\frac{2(1+\log n)}{n}} + B\sqrt{\frac{2}{n}\log\frac{1}{\delta}} + \frac{1-\rho}{\rho}M$$

- $\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D}[\ell(h(\mathbf{x};\hat{\mathbf{w}}),\mathbf{y})]$: risk of the approximately leaned model
- $\frac{1}{n}\sum_{t=1}^{n} r_t(\mathbf{w}^*)$: empirical risk of the model with **exact** learning
- ρ: approximation ratio.

Risk bound, learning submodular mixtures with subgradient decent

Theorem (Lin & Bilmes 2012 (summarized))

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D}[\ell(h(\mathbf{x};\hat{\mathbf{w}}),\mathbf{y})] \leq \frac{1}{\rho}\left(\frac{1}{n}\sum_{t=1}^{n} r_t(\mathbf{w}^*)\right) + S(n),$$

where

$$S(n) = \frac{MG}{\rho}\sqrt{\frac{2(1+\log n)}{n}} + B\sqrt{\frac{2}{n}\log\frac{1}{\delta}} + \frac{1-\rho}{\rho}M$$

- $\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D}[\ell(h(\mathbf{x};\hat{\mathbf{w}}),\mathbf{y})]$: risk of the approximately leaned model
- $\frac{1}{n}\sum_{t=1}^{n} r_t(\mathbf{w}^*)$: empirical risk of the model with **exact** learning
- ρ: approximation ratio.

Risk bound, learning submodular mixtures with subgradient decent

Theorem (Lin & Bilmes 2012 (summarized))

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D}[\ell(h(\mathbf{x};\hat{\mathbf{w}}),\mathbf{y})] \leq \frac{1}{\rho}\left(\frac{1}{n}\sum_{t=1}^{n} r_t(\mathbf{w}^*)\right) + S(n),$$

where

$$S(n) = \frac{MG}{\rho}\sqrt{\frac{2(1+\log n)}{n}} + B\sqrt{\frac{2}{n}\log\frac{1}{\delta}} + \frac{1-\rho}{\rho}M$$

- $\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D}[\ell(h(\mathbf{x};\hat{\mathbf{w}}),\mathbf{y})]$: risk of the approximately leaned model
- $\frac{1}{n}\sum_{t=1}^{n} r_t(\mathbf{w}^*)$: empirical risk of the model with **exact** learning
- ρ: approximation ratio.

Risk bound, learning submodular mixtures with subgradient decent

Theorem (Lin & Bilmes 2012 (summarized))

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D}[\ell(h(\mathbf{x};\hat{\mathbf{w}}),\mathbf{y})] \leq \frac{1}{\rho}\left(\frac{1}{n}\sum_{t=1}^{n} r_t(\mathbf{w}^*)\right) + S(n),$$

where

$$S(n) = \frac{MG}{\rho}\sqrt{\frac{2(1+\log n)}{n}} + B\sqrt{\frac{2}{n}\log\frac{1}{\delta}} + \frac{1-\rho}{\rho}M$$

- $\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D}[\ell(h(\mathbf{x};\hat{\mathbf{w}}),\mathbf{y})]$: risk of the approximately leaned model
- $\frac{1}{n}\sum_{t=1}^{n} r_t(\mathbf{w}^*)$: empirical risk of the model with **exact** learning
- ρ: approximation ratio.

Risk bound, learning submodular mixtures with subgradient decent

Theorem (Lin & Bilmes 2012 (summarized))

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D}[\ell(h(\mathbf{x};\hat{\mathbf{w}}),\mathbf{y})] \leq \frac{1}{\rho}\left(\frac{1}{n}\sum_{t=1}^{n} r_t(\mathbf{w}^*)\right) + S(n),$$

where

$$S(n) = \frac{MG}{\rho}\sqrt{\frac{2(1+\log n)}{n}} + B\sqrt{\frac{2}{n}\log\frac{1}{\delta}} + \frac{1-\rho}{\rho}M$$

- $\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D}[\ell(h(\mathbf{x};\hat{\mathbf{w}}),\mathbf{y})]$: risk of the approximately leaned model
- $\frac{1}{n}\sum_{t=1}^{n} r_t(\mathbf{w}^*)$: empirical risk of the model with **exact** learning
- ρ: approximation ratio. $\rho \approx 1$ for greedy algorithm in budgeted submodular maximization

Submodular Components for Document Summarization

Components:

- Coverage (fidelity) components:

$$f_\beta(S) = \frac{1}{|V|} \sum_{i \in V} \min \left\{ \frac{\mathcal{C}_i(S)}{\mathcal{C}_i(V)}, \beta \right\},$$

Submodular Components for Document Summarization

Components:

- Coverage (fidelity) components:

$$f_\beta(S) = \frac{1}{|V|} \sum_{i \in V} \min \left\{ \frac{\mathcal{C}_i(S)}{\mathcal{C}_i(V)}, \beta \right\},$$

- Diversity components, for partition $\mathcal{P} = (P_1, \ldots, P_K)$:

$$f_{\mathcal{P},\alpha}(S) = \frac{\sum_{k=1}^{K} \left(\sum_{i \in S \cap P_k} r_i \right)^\alpha}{\sum_{k=1}^{K} \left(\sum_{i \in P_k} r_i \right)^\alpha},$$

Submodular Components for Document Summarization

Components:

- Coverage (fidelity) components:

$$f_\beta(S) = \frac{1}{|V|} \sum_{i \in V} \min \left\{ \frac{\mathcal{C}_i(S)}{\mathcal{C}_i(V)}, \beta \right\},$$

- Diversity components, for partition $\mathcal{P} = (P_1, \ldots, P_K)$:

$$f_{\mathcal{P},\alpha}(S) = \frac{\sum_{k=1}^{K} \left(\sum_{i \in S \cap P_k} r_i \right)^\alpha}{\sum_{k=1}^{K} \left(\sum_{i \in P_k} r_i \right)^\alpha},$$

- Clustered facility location like components, for partition $\mathcal{P}$

$$f_{\mathcal{P}}(S) = \frac{1}{K} \sum_{k=1}^{K} \max_{i \in S \cap P_k} r_i, \tag{21}$$

NIST's ROUGE-N evaluation function

NIST's ROUGE-N score is the standard evaluation measure, and it is polymatroidal:

$$f_{\text{ROUGE-N}}(S) \triangleq \frac{\sum_{i=1}^{K} \sum_{e \in R_i} \min(c_e(S), r_{e,i})}{\sum_{i=1}^{K} \sum_{e \in R_i} r_{e,i}},$$

where

- S is the candidate summary (a set of sentences extracted from the ground set V)
- $c_e : 2^V \rightarrow \mathbb{Z}_+$ is the number of times an n-gram e occurs in summary S, clearly a modular function for each e.
- R_i is the set of n-grams contained in the reference summary i (given K reference summaries).
- and $r_{e,i}$ is the number of times n-gram e occurs in reference summary i.
- Note again, ROUGE-N is unavailable to optimize directly.

Loss Function ℓ

- ROUGE-N can't be used since it measures "accuracy" rather than loss.

Loss Function ℓ

- ROUGE-N can't be used since it measures "accuracy" rather than loss.
- $1 -$ ROUGE-N is supermodular, sum would require a submodular-supermodular procedure (SSP) Narasimhan&Bilmes 2005 (demonstrates a need for good quality SSP).

Loss Function ℓ

- ROUGE-N can't be used since it measures "accuracy" rather than loss.
- 1 − ROUGE-N is supermodular, sum would require a submodular-supermodular procedure (SSP) Narasimhan&Bilmes 2005 (demonstrates a need for good quality SSP).
- Surrogate loss: we use a form of "complement recall", of the form:

$$\ell_{\text{ROUGE}}(S) \triangleq \frac{\sum_{e \in \bar{R}} \min(c_e(S), r_e)}{\sum_{e \in \bar{R}} r_e}, \tag{22}$$

where

$$\bar{R} = N \backslash \bigcup_i R_i, \tag{23}$$

and where N is the set of all the n-grams occur in the documents, and r_e is the number of times n-gram e occurs in the documents.

Loss Function ℓ

- ROUGE-N can't be used since it measures "accuracy" rather than loss.
- 1 − ROUGE-N is supermodular, sum would require a submodular-supermodular procedure (SSP) Narasimhan&Bilmes 2005 (demonstrates a need for good quality SSP).
- Surrogate loss: we use a form of "complement recall", of the form:

$$\ell_{\text{ROUGE}}(S) \triangleq \frac{\sum_{e \in \bar{R}} \min(c_e(S), r_e)}{\sum_{e \in \bar{R}} r_e}, \tag{22}$$

where

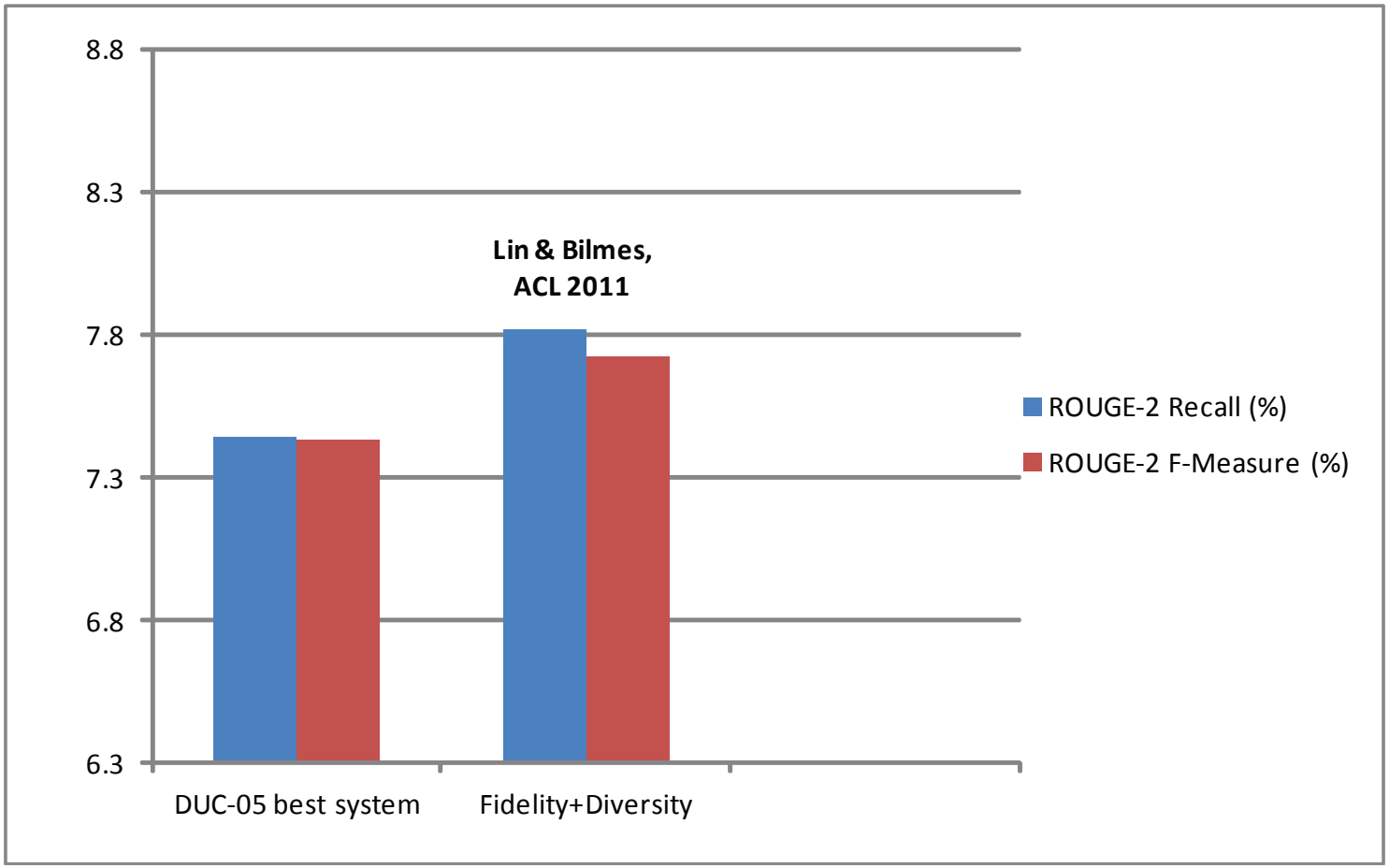
$$\bar{R} = N \backslash \bigcup_i R_i, \tag{23}$$

and where N is the set of all the n-grams occur in the documents, and r_e is the number of times n-gram e occurs in the documents.

- ℓ_{ROUGE} is clearly polymatroidal.

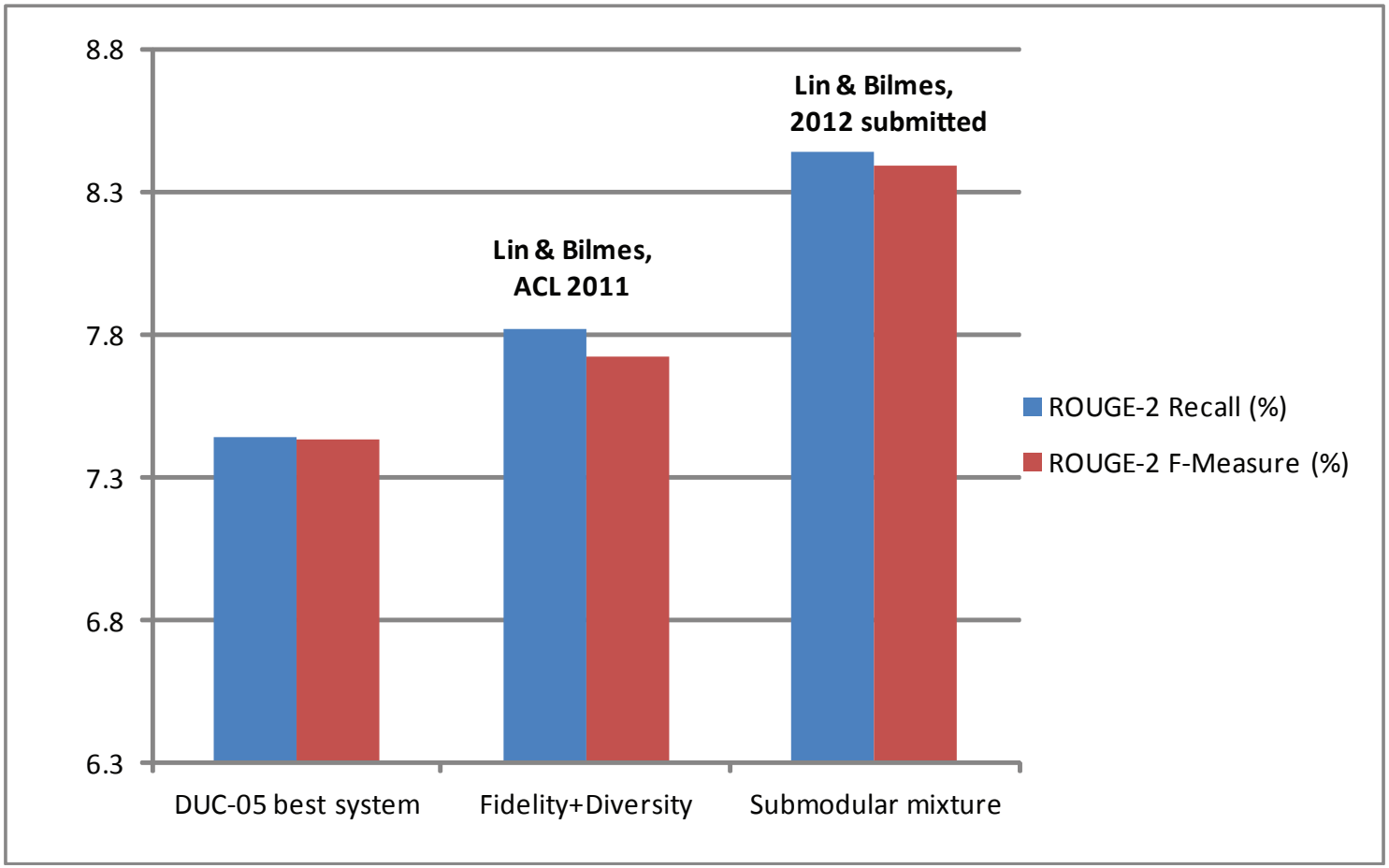
Query-focused Summarization Results

DUC-05: DUC-06 and DUC-07 were used in submodular mixture learning



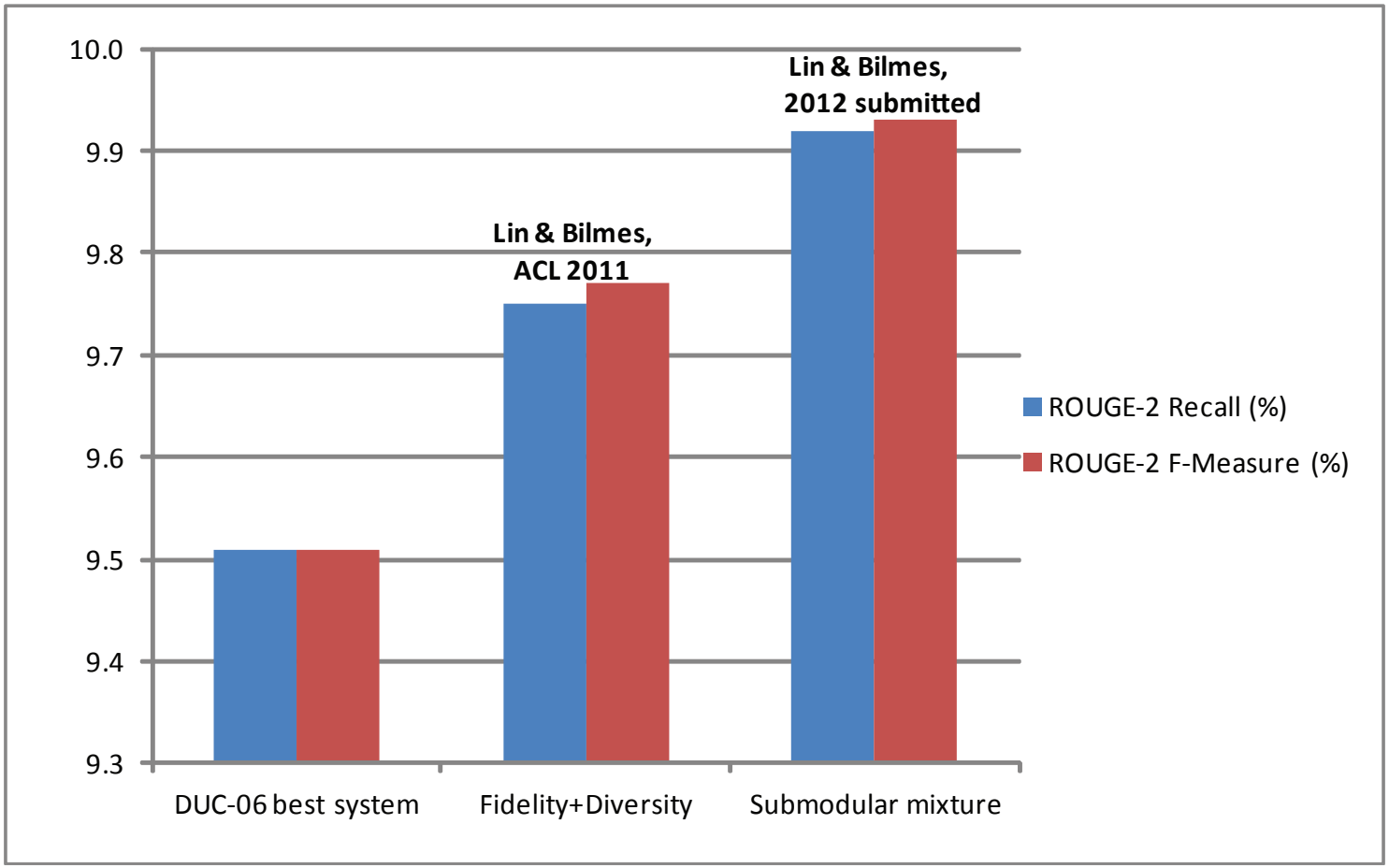
Query-focused Summarization Results

DUC-05: DUC-06 and DUC-07 were used in submodular mixture learning



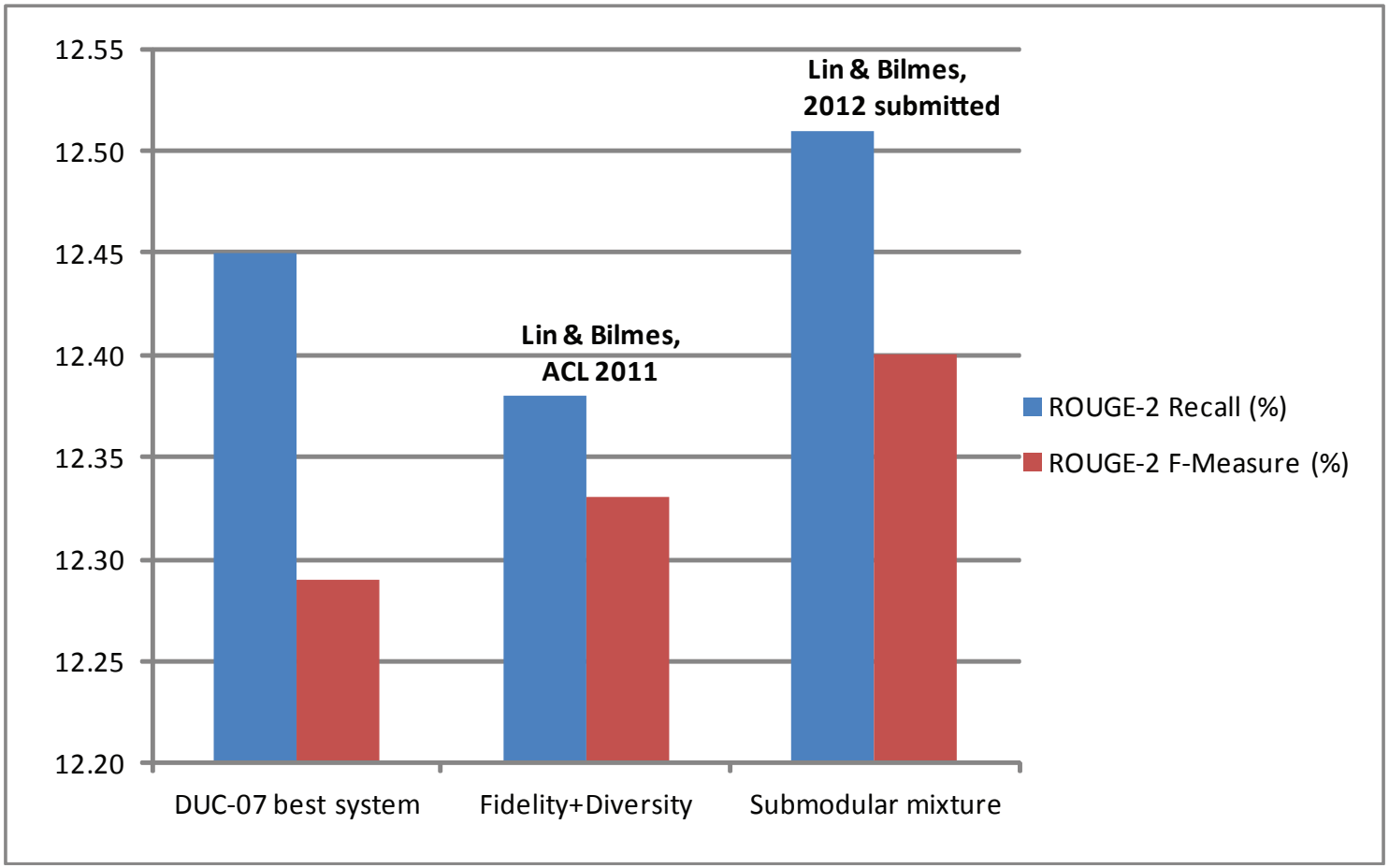
Query-focused Summarization Results

DUC-06: DUC-05 and DUC-07 were used in submodular mixture learning

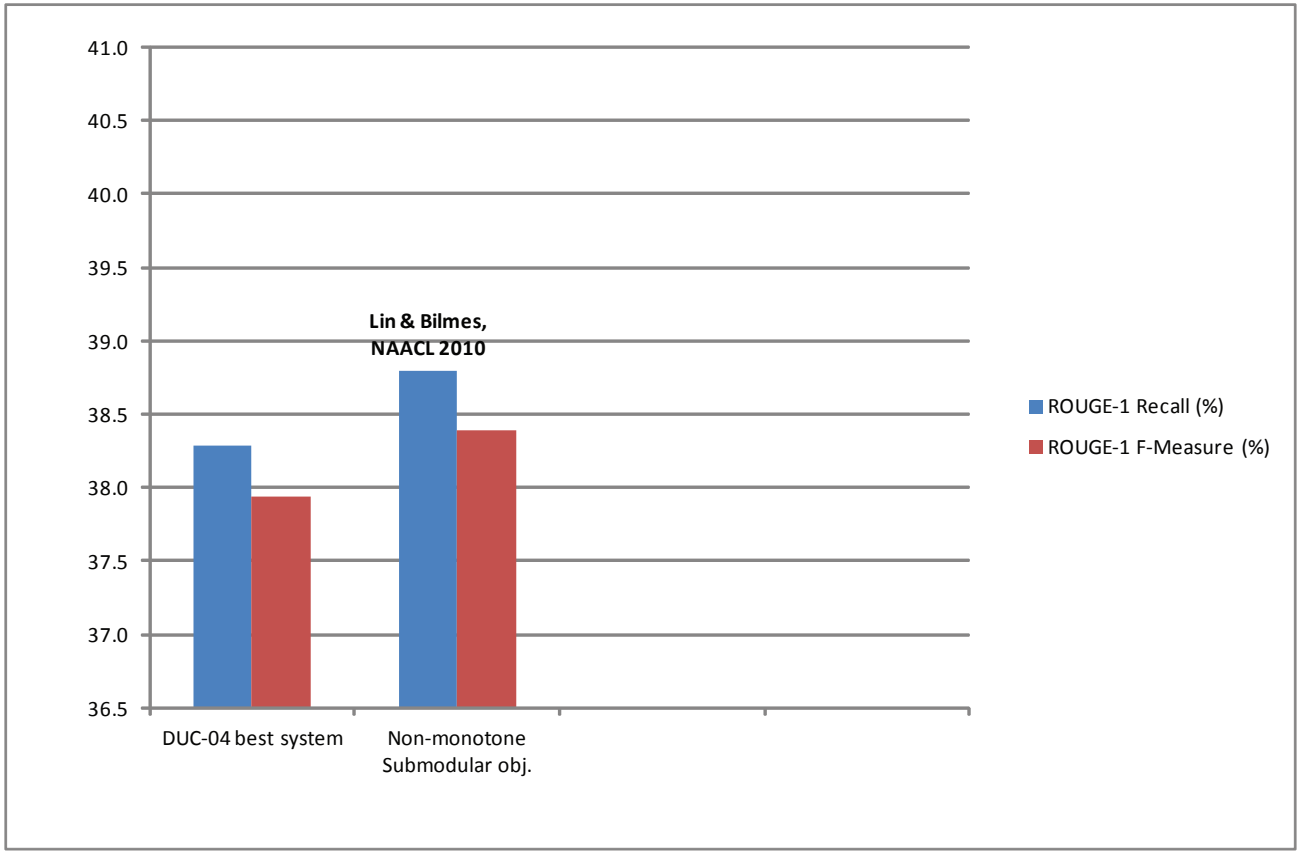


Query-focused Summarization Results

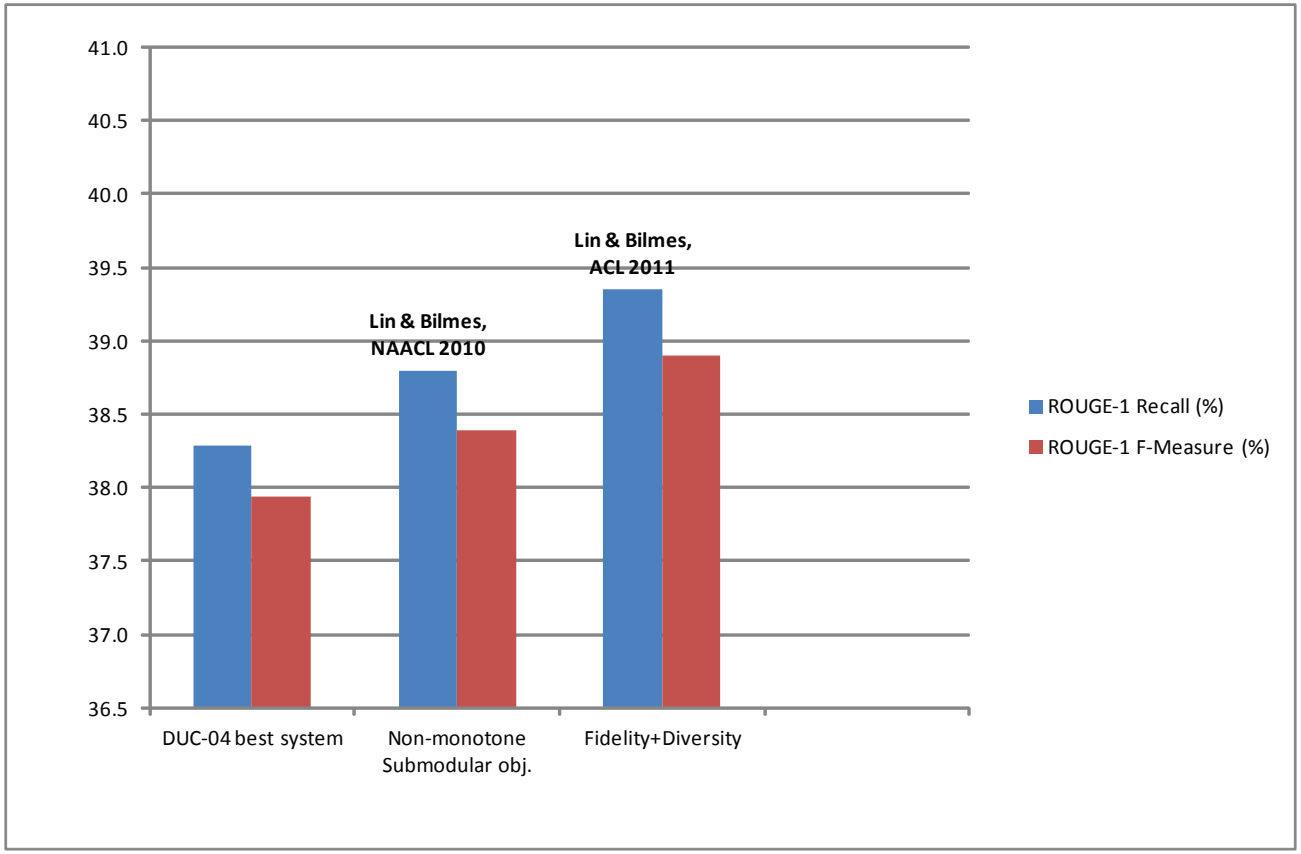
DUC-07: DUC-05 and DUC-06 were used in submodular mixture learning



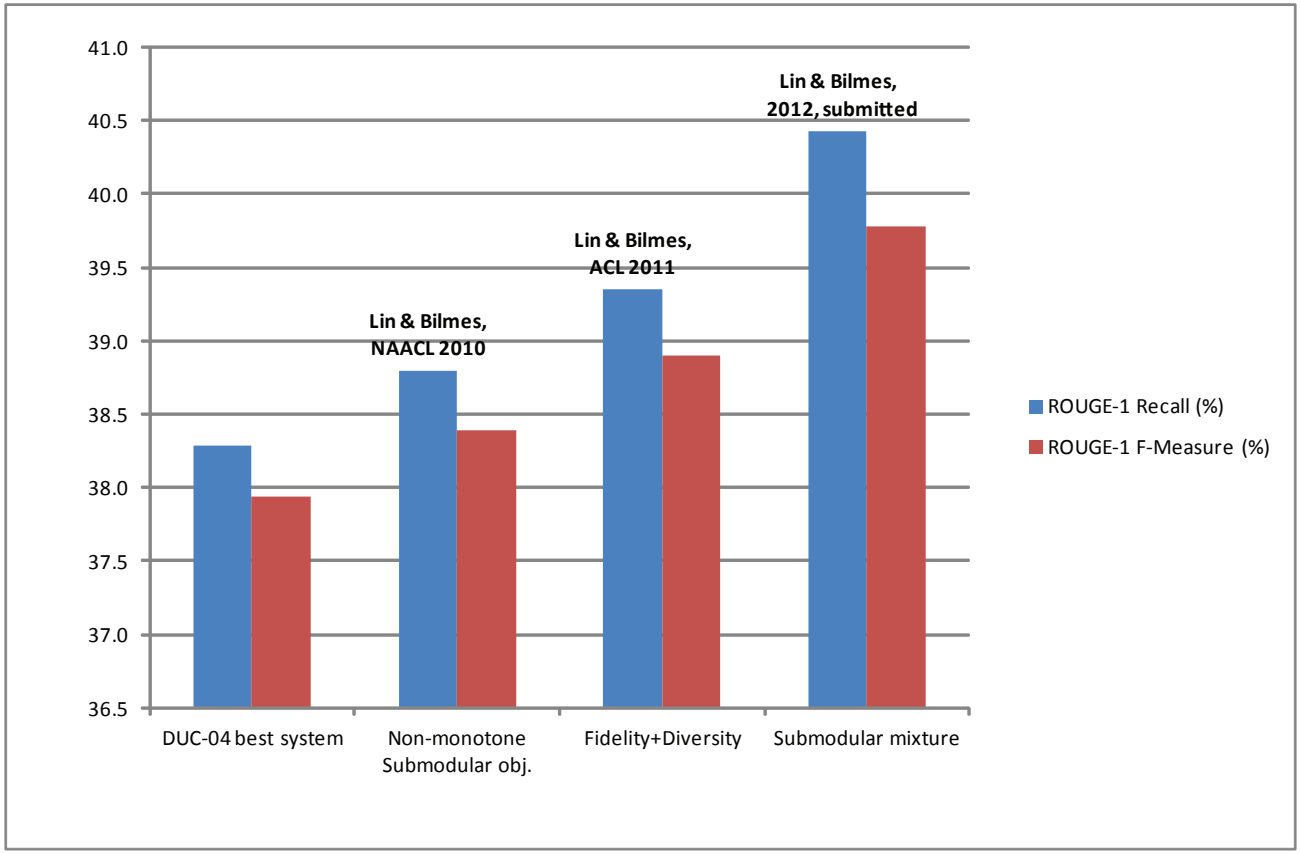
Generic Summarization Results



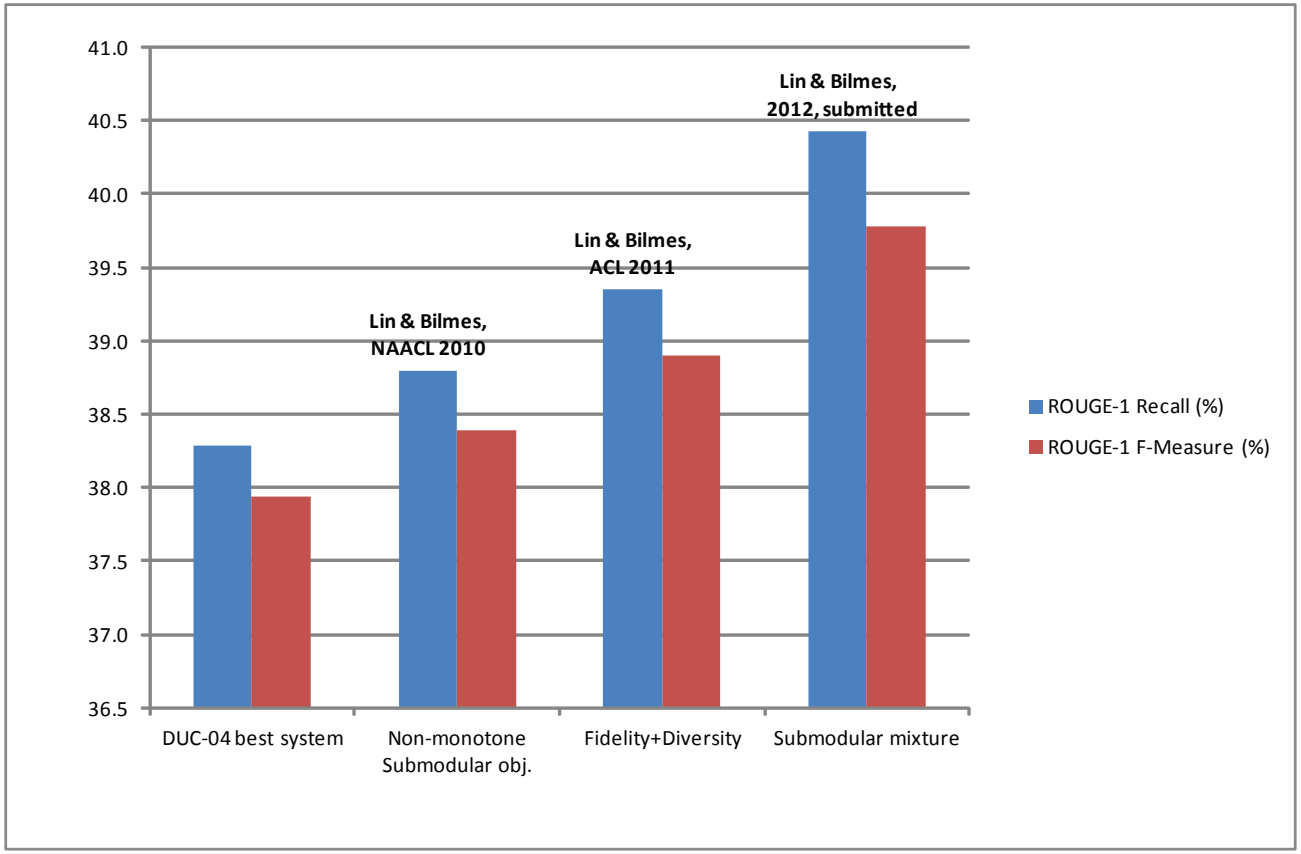
Generic Summarization Results



Generic Summarization Results



Generic Summarization Results



- In general, the best results ever reported on DUC-04, 05, 06 and 07 (first reported here).

Outline

Outline

Active Learning and Transductive Semi-Supervised Learning

- Batch/Offline active learning: Given a set V of unlabeled data items, the learner must choose a subset $L \subseteq V$ of the items that are to be labeled (and learnt from).

Active Learning and Transductive Semi-Supervised Learning

- Batch/Offline active learning: Given a set V of unlabeled data items, the learner must choose a subset $L \subseteq V$ of the items that are to be labeled (and learnt from).
- Transductive Semi-Supervised Learning: Given a subset L of data items that are already labeled, deduce the labels of all remaining items $V \setminus L$ without using any additional labels.

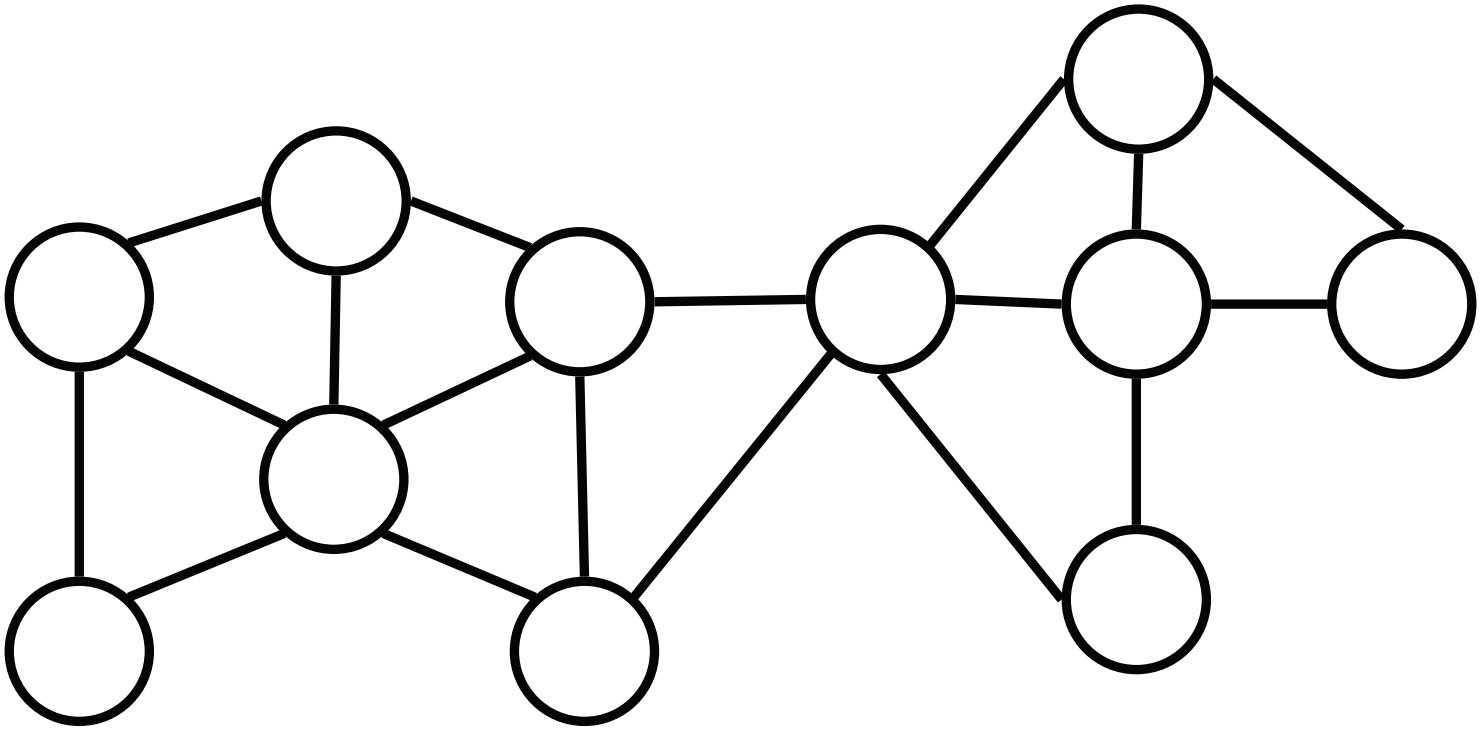
Active Learning and Transductive Semi-Supervised Learning

- Batch/Offline active learning: Given a set V of unlabeled data items, the learner must choose a subset $L \subseteq V$ of the items that are to be labeled (and learnt from).
- Transductive Semi-Supervised Learning: Given a subset L of data items that are already labeled, deduce the labels of all remaining items $V \setminus L$ without using any additional labels.
- Ideally do both, with a bound on the completion error.

Active Learning and Transductive Semi-Supervised Learning

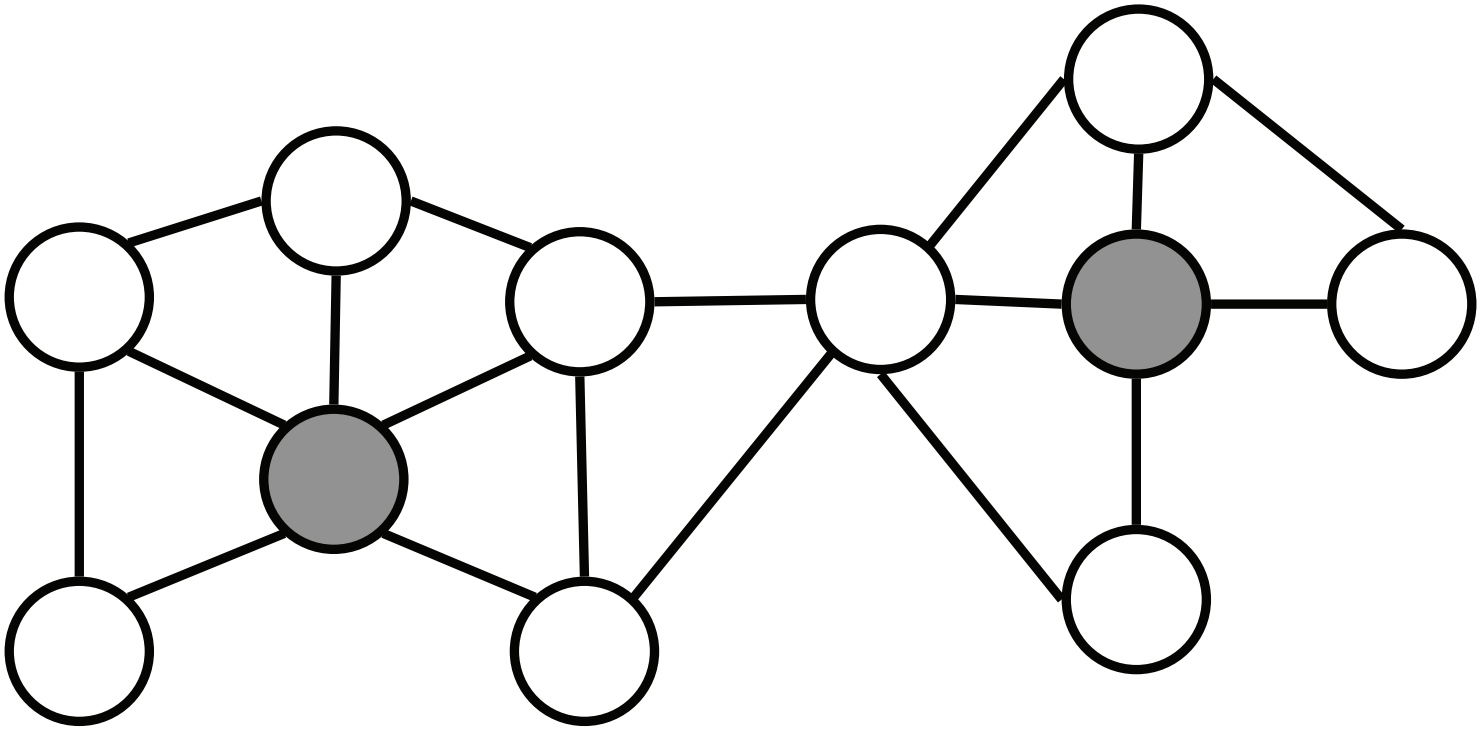
- Batch/Offline active learning: Given a set V of unlabeled data items, the learner must choose a subset $L \subseteq V$ of the items that are to be labeled (and learnt from).
- Transductive Semi-Supervised Learning: Given a subset L of data items that are already labeled, deduce the labels of all remaining items $V \setminus L$ without using any additional labels.
- Ideally do both, with a bound on the completion error.
- Very little work so far on methods that can do both.

Given Unlabeled Data

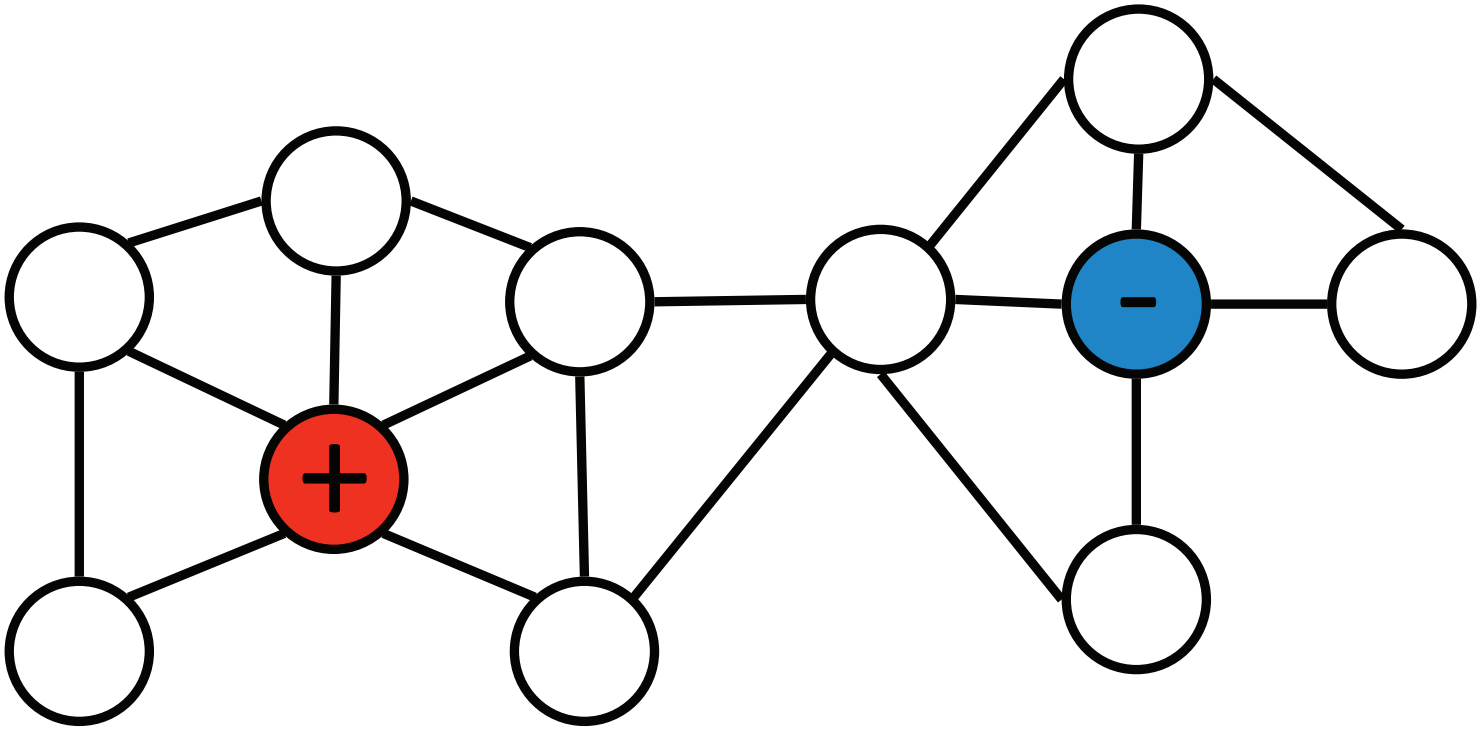


For example, as represented by graph

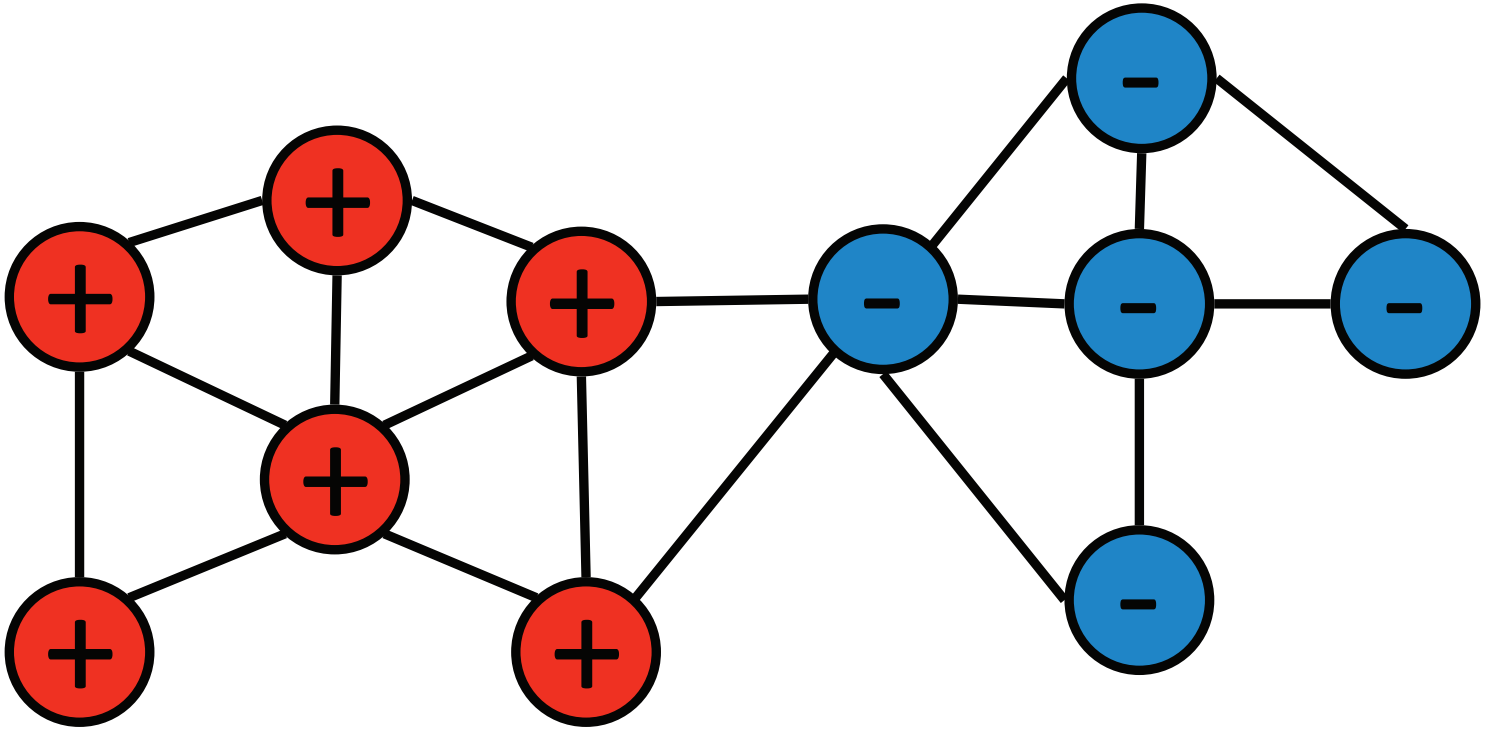
Learner chooses a labeled set $L \subseteq V$



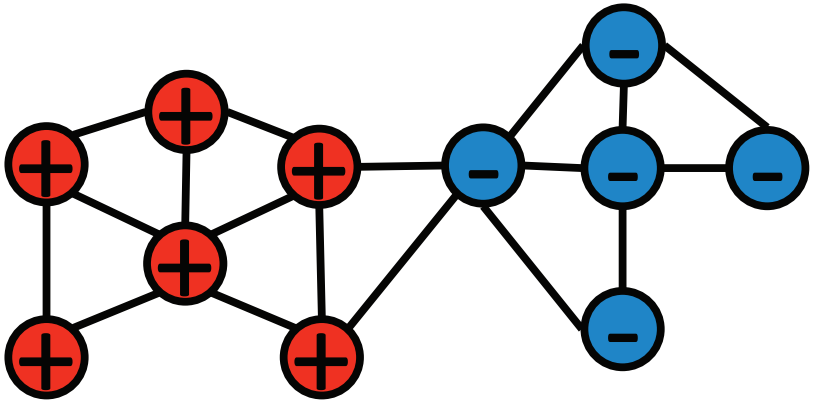
Nature reveals labels $y_L \in \{0,1\}^L$



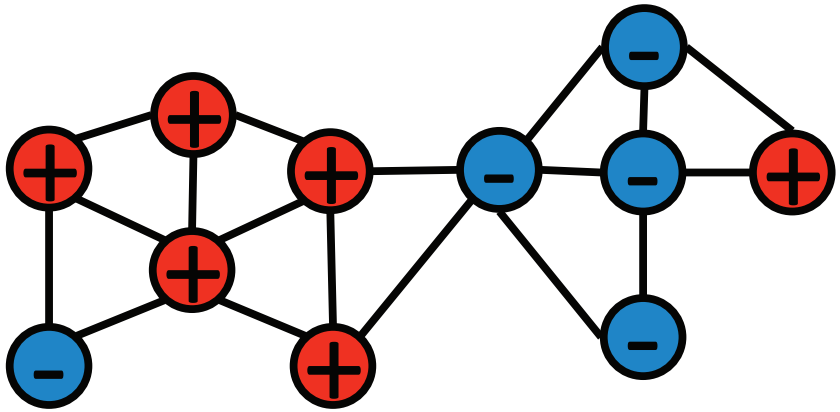
Learner predicts labels $\hat{y} \in \{0,1\}^V$



Learner suffers loss $\|\hat{y} - y\|_1$



Predicted



Actual

$$\|\hat{y} - y\|_1 = 2 \tag{24}$$

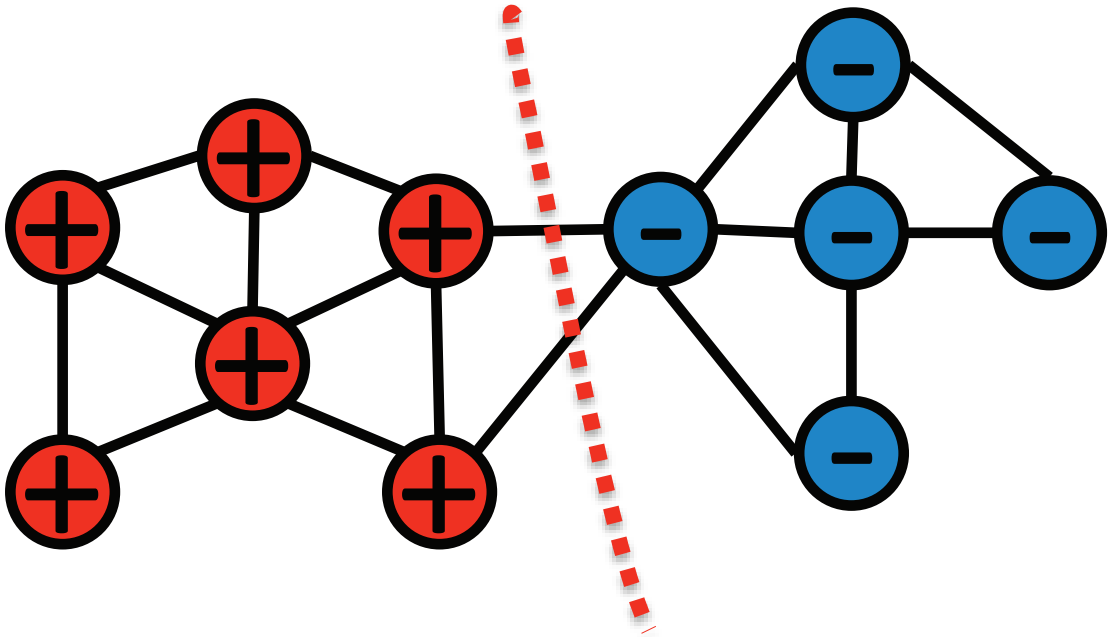
Basic Questions

- What should we assume about y?
- How should we predict $\hat{y}$ using y_L?
- How should we select L?
- How can we bound the error?

Outline

Learning on graphs

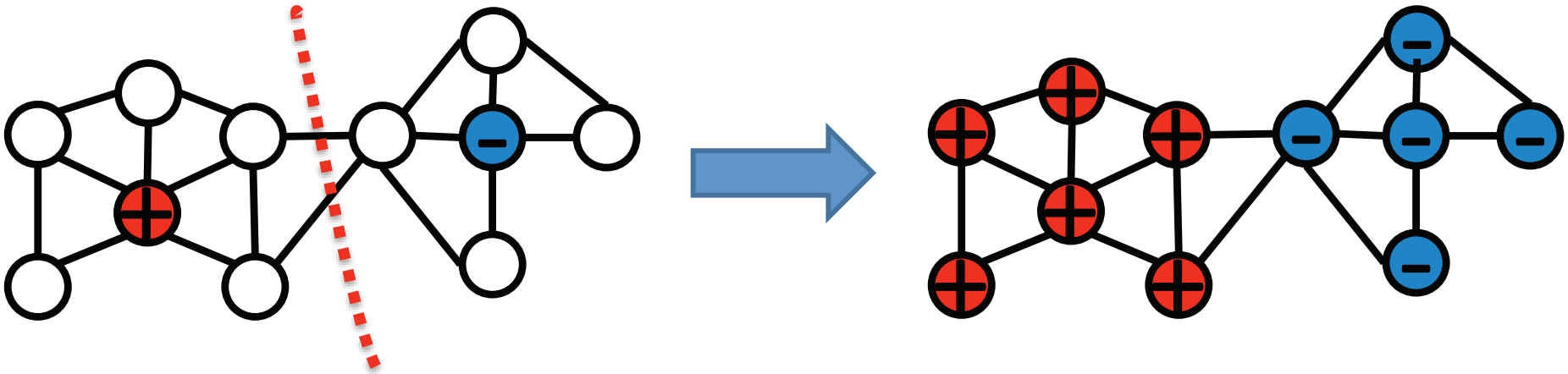
- What should we assume about y?
- Standard assumption: small cut value
- I.e., $\Phi(y) = \sum_{i<j}(y_i - y_j)^2 w_{ij}$ is small, where w_{ij} measures similarity between item i and j.
- This can be seen as exploiting "smoothness" assumption.



$\Phi(y) = 2$

Prediction on graphs

- How should we predict $\hat{y}$ using y_L?
- A standard approach: min-cut (Blum & Chawla 2001)
- Choose $\hat{y}$ to minimize $\Phi(\hat{y})$ s.t. $\hat{y}_L = y_L$
- Reduces to a standard min-cut computation
- This can be seen as a "smoothness" assumption about nature.



Active learning on graphs

- How should we select L?
- In previous work, we proposed the following objective

$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{25}$$

where $\Gamma(T)$ is the cut value between T and $V \setminus T$.

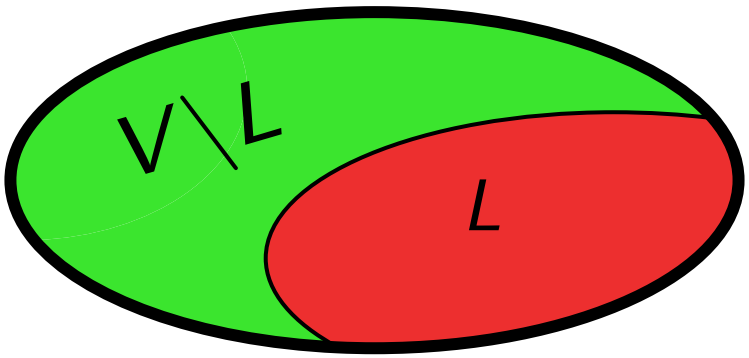
Active learning on graphs

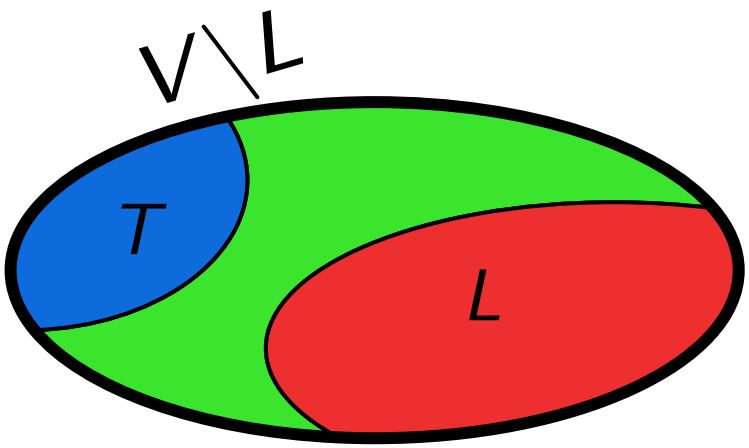
- How should we select L?
- In previous work, we proposed the following objective

$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{25}$$

where $\Gamma(T)$ is the cut value between T and $V \setminus T$.

- Small $\Psi(L)$ means an adversary can cut away many points from L without cutting many edges.





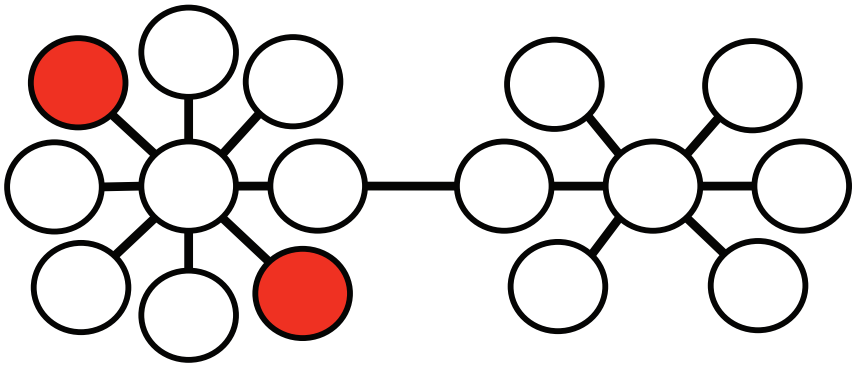
Active learning on graphs

- How should we select L?
- In previous work, we proposed the following objective

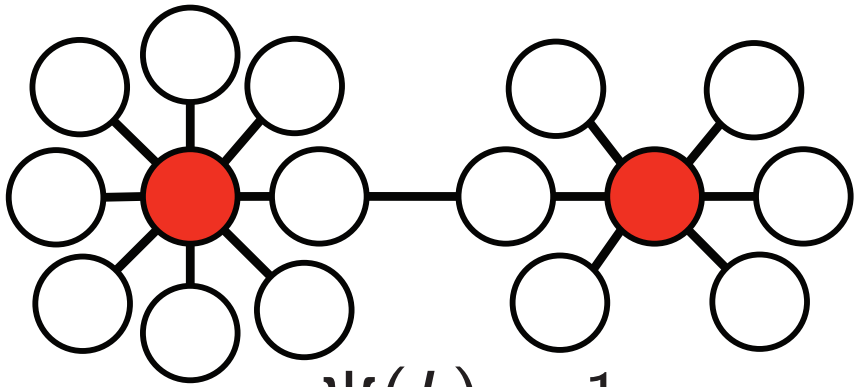
$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{25}$$

where $\Gamma(T)$ is the cut value between T and $V \setminus T$.

- Small $\Psi(L)$ means an adversary can cut away many points from L without cutting many edges.



$\Psi(L) = 1/8$



$\Psi(L) = 1$

Active learning on graphs

- How can we bound the error?

Theorem

Guillory & Bilmes 2009 Assume $\hat{y}$ minimizes $\Phi(\hat{y})$ subject to $\hat{y}_L = y_L$. Then

$$\|\hat{y} - y\|_1 \leq 2\frac{\Phi(y)}{\Psi(L)} \tag{26}$$

- Intuition: $\textit{Error} \leq \frac{\text{Complexity of true labels}}{\text{Quality of labeled set}}$
- Note: Deterministic bound, holds for adversarial labels.

Drawbacks of previous work

- Restricted to only graph based, min-cut learning.
- Not clear how to efficiently maximize $\Psi(L)$
  - Can compute in polynomial time (Guillory & Bilmes, 2009)
  - Only heuristic methods known for maximizing in general case.
- Not guaranteed that this bound is the right bound

More recent contributions

- Guillory & Bilmes, UAI 2011.
- A new more general bound on error, parameterized by an arbitrarily chosen submodular function.
- An active, semi-supervised learning method for approximately minimizing this bound.
- Proof that minimizing this bound exactly is NP-hard
- Theoretical evidence that this is the "right" bound.

Outline

Submodular Functions For Learning

- $\Gamma(T)$ (cut value) is symmetric and submodular
- This makes $\Gamma(T)$ "nice" for learning
  - Easy to analyze
  - Can minimize exactly in polynomial time
- For other learning settings, other symmetric submodular functions make sense
  - Hypergraph cut is symmetric submodular
  - Symmetric mutual information is symmetric and submodular
  - An arbitrary submodular function F (e.g., matroid rank) can be symmeterized

$$\Gamma(S) = F(S) + F(V \setminus S) - F(V) \tag{27}$$

Generalized Error Bound

Theorem

For any symmetric submodular $\Gamma(S)$, assume $\hat{y}$ minimizes $\Phi(\hat{y})$ subject to $\hat{y}_L = y_L$. Then

$$\|\hat{y} - y\|_1 \leq 2\frac{\Phi(y)}{\Psi(L)} \tag{28}$$

- Here, Φ and Ψ are defined in terms of the symmetric submodular function Γ, not graph cut.

$$\Phi(y) = \Gamma(V_{y=1}) \text{ and } \Psi(S) = \min_{T \subseteq V \setminus S : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{29}$$

- Each choice of Γ gives different error bound (objective is "parameterized" by a submodular function).
- Minimizing $\Phi(\hat{y})$ s.t. $\hat{y}_L = y_L$ can be done in polynomial time (submodular function minimization).

Can we efficiently maximize Ψ?

- Two related problems
  1. Maximize $\Psi(L)$ subject to $|L| \leq k$
  2. Minimize $|L|$ subject to $\Psi(L) \geq \lambda$
- If $\Psi(L)$ were submodular, we could use well-known results of greedy procedure
  - $(1 - 1/e)$-approximation to 1. (Nemhauser et al. 1978)
  - $1 + \ln F(V)$ approximation for 2. (Wolsey 1981) for integer valued F
- Unfortunately, $\Psi(L)$ is not submodular.

Approximation result

- Define a surrogate objective $F_\lambda(S)$ s.t.

$$F_\lambda(S) = \min_{T \subseteq V \setminus S : T \neq \emptyset} \Gamma(T) - \lambda |T| \tag{30}$$

- $F_\lambda(S)$ is monotone non-decreasing submodular
- Evaluating $F_\lambda(S)$ at S requires SFM.
- $F_\lambda(S) \geq 0$ iff $\Psi(S) \geq \lambda$
- Can then use standard methods for $F_\lambda(S)$.

Theorem

For any integral symmetric submodular function $\Gamma(S)$, integer λ, greedily maximizing $F_\lambda(L)$ gives L with $\Phi(L) \geq \lambda$, and $|L| \leq (1 + \ln \lambda) \min_{L:\Phi(L)\geq\lambda} |L|$

Can we do better?

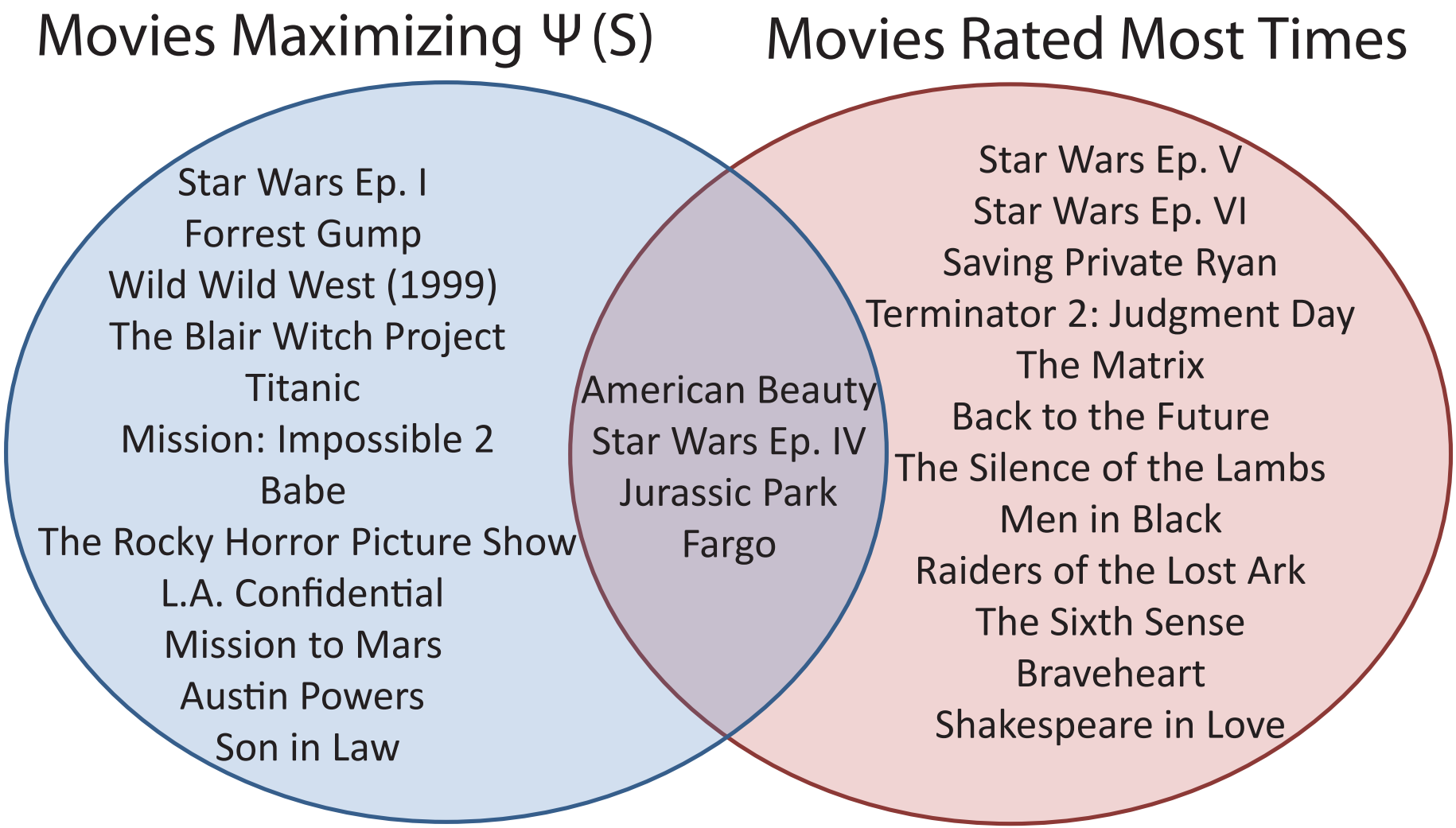
- Is it possible to maximize $\Psi(L)$ exactly?
- Probably not, we show the problem is NP-complete
  - Holds also if $\Gamma(S)$ is even the cut function
  - Reduction from vertex cover on fixed degree graphs.
- Is there a strictly better bound?
- Not of the same form
  - No function larger than $\Psi(L)$ for which the bound of this form holds.
  - Suggests this is the "right" bound.

Outline

Experiments: Movie Recommendation

- Which movies would a user rate to get accurate recommendations from collaborative filtering?
- We pose this problem as active learning over a hypergraph encoding user preferences using $\Gamma(S)$ set to hypergraph cut.
- Two hypergraph edges for each user.
  - Hypergraph edge connecting all movies a user likes
  - Hypergraph edge connecting all movies a user dislikes
- Partitions with low hypergraph cut value are consistent (on average) with user preferences.

Results on Movielens data



Outline

Summary

- Submodular functions are finding ever more application in machine learning.
- They naturally represent and successfully solve the problem of document summarization.
- They can be used to parameterize a class of joint active/semi-supervised learning algorithms.
- A need for both fast submodular function maximization and minimization on large ground set sizes.